Interactive Visualization of MARS Spectral CT Datasets

A thesis submitted in fulfilment of the
requirements for the degree of
Doctor of Philosophy
by
Veera Bhadra Harish Mandalika

University of Canterbury
2017

To my parents Kantharao and Subhadra for their love, support, and encouragement in putting me through the best education possible. This thesis would certainly not have existed without them.

To my wife Ramya for her unending support, patience, and love. I wouldn't have gotten through this doctorate if it wasn't for her.

To my late grandfather Viswanadha Sastry Kadiyala, my late grandmother Lakshmi Sodemma, my late uncle Sarveshwara Lakshmi Narasimha Somayajulu and my family for their unconditional love throughout my life.

Abstract

This thesis develops a 3D manipulation approach for MARS spectral CT datasets for medical diagnosis/imaging. The key outcome is the design of a novel 2D/3D hybrid user interface for interactive exploration. A study is presented that shows the effectiveness of the hybrid user interface compared to a standard 2D interface for the novice (medical student) as well as experienced (radiology resident) users in diagnostic radiology. The study demonstrated that the hybrid interface is an effective approach that requires minimal training to achieve consistently accurate results. In fact, using the hybrid interface, the students' accuracy scores matched that of trained residents.

Spectral CT (also referred to as colour CT) is an emerging medical imaging modality that acquires data over multiple x-ray colours in order to provide more accurate images and expand medical applications. The MARS project has developed a small specimen spectral CT scanner for pre-clinical research. This thesis work builds on MARS Vision, an in-house visualisation tool developed for analysing MARS datasets.

The first accomplishments in this thesis advance the tools and features in MARS Vision. Some of the main features include stereoscopic 3D rendering, rapid mesh extraction, a rendering engine, and an arbitrary slice view. These features also form a framework to facilitate primary research into 3D manipulation.

The thesis presents a hybrid user interface that combines the zSpace stereoscopic display (with 3D stylus input) with a standard 2D display (with mouse and keyboard input). The interface augments the diagnostic radiology workflow by adding a 3D component, as opposed to replacing the existing workflow.

This thesis also presents an evaluation of the hybrid interface by comparing it to a standard 2D interface (based on Inteleviewer) along with a 3D only interface. The study involved 21 medical students and 10 radiology residents performing a scoliosis diagnosis task using the three interfaces. It demonstrated that the hybrid interface was an effective alternative for 3D manipulation in medical diagnosis. This interface is implemented in the commercial version of MARS Vision and is currently being used by several pre-clinical research groups worldwide.

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit.

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Publication:

Veera Bhadra Harish Mandalika, Alexander I. Chernoglazov, Mark Billinghurst, Christoph Bartneck, Michael A. Hurrell, Niels de Ruiter, Anthony P. H. Butler, Philip H. Butler. *A Hybrid 2D/3D User Interface for Radiological Diagnosis*, Published in the Journal of Digital Imaging (2017).

Please detail the nature and extent (%) of contribution by the candidate:

For this publication, I developed a hybrid 2D/3D user interface for medical data exploration. I also designed and carried out a user study to evaluate my hybrid interface against the existing 2D radiology software. I analysed the results and presented them along with my hybrid interface design in the paper. Contribution – 70%

Certification by Co-authors:

If there is more than one co-author then a single co-author can sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the candidate was the lead author of the co-authored work he or she wrote the text

Name: *Alexander Chernoglazov* Signature:  Date: *19/10/2017*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit.

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Publication:

R. Aamir, A. Chernoglazov, C. J. Bateman, A. P. H. Butler, P. H. Butler, N. G. Anderson, S. T. Bell, R. Panta, J. L. Healy, J. L. Mohr, K. Rajendran, M. F. Walsh, N. J. A. de Ruiter, S. P. Gieseg, T. Woodfield, P. F. Renaud, L. Brooke, S. Abdul-Majid, M. Clyne, R. Glendenning, P. J. Bones, M. Billinghurst, C. Bartneck, **H. Mandalika**, R. Grasset, N. Schleich, N. Scott, S. J. Nik, A. Opie, T. Janmale, D. N. Tang, D. Kim, R. M. Doesburg, R. Zainon, J. P. Ronaldson, N. J. Cook, D. J. Smithies, K. Hodge (2014), *MARS spectral molecular imaging of lamb tissue: data collection and image analysis, Journal of Instrumentation*, Vol. 9, Num. 2.

Please detail the nature and extent (%) of contribution by the candidate:

For this publication, I helped develop the software MARS Vision, which was used to produce images to present the distribution of materials in a piece of lamb meat scanned in the MARS scanner. My work primarily consisted of implementing viewpoint camera manipulation. Contribution – 5%

Certification by Co-authors:

If there is more than one co-author then a single co-author can sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the candidate was the lead author of the co-authored work he or she wrote the text

Name: *D r Aamir Younis Raja* Signature:  Date: *19 Oct 2017*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit.

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Publication:

K. Rajendran, M. F. Walsh, N. J. A. de Ruiter, A. I. Chernoglazov, R. K. Panta, A. P. H. Butler, P. H. Butler, S. T. Bell, N. G. Anderson, T. B. F. Woodfield, S. J. Tredinnick, J. L. Healy, C. J. Bateman, R. Aamir, R. M. N. Doesburg, P. F. Renaud, S. P. Gieseg, D. J. Smithies, J. L. Mohr, **V. B. H. Mandalika**, A. M. T. Opie, N. J. Cook, J. P. Ronaldson, S. J. Nik, A. Atharifard, M. Clyne, P. J. Bones, C. Bartneck, R. Grasset, N. Schleich, M. Billinghurst. (2014), *Reducing beam hardening effects and metal artefacts in spectral CT using Medipix3RX*, Journal of Instrumentation, Vol. 9, Num. 3.

Please detail the nature and extent (%) of contribution by the candidate:

For this publication, I helped develop the software MARS Vision, which was used to produce images for 3D visualisation that demonstrated the reduction in beam hardening effects through narrow energy bins which is possible using the MARS scanner. My work consisted of implementing effective viewpoint and light position manipulation controls. Contribution – 4%

Certification by Co-authors:

If there is more than one co-author then a single co-author can sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the candidate was the lead author of the co-authored work he or she wrote the text

Name: Kishore Rajendran, Ph.D. Signature: R. Kishore Date: 18 October 2017

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit.

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Publication:

A. Atharifard, J. L. Healy, B. P. Goulter, M. Ramyar, L. Vanden Broeke, M. F. Walsh, C. C. Onyema, R. K. Panta, R. Aamir, D. J. Smithies, R. Doesburg, M. Anjomrouz, M. Shamshad, S. Bheesettea, K. Rajendran, N. J. A. de Ruiter, D. Knight, A. Chernoglazov, **H. Mandalika**, S. T. Bell, C. J. Bateman, A. P. H. Butler, P. H. Butler, *Per-pixel energy calibration of photon counting detectors*, Journal of Instrumentation, Vol. 12, Num. 3.

Please detail the nature and extent (%) of contribution by the candidate:

For this paper, I was a consultant for knowledge transfer. I participated in the discussions on the implication of per-pixel energy calibration on the quality of visualisation. It is expected that this technique will be integrated into the MARS imaging and will improve the quality of image processing and thereby improve the quality of MARS data visualisation. Contribution – 8 %

Certification by Co-authors:

If there is more than one co-author then a single co-author can sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the candidate was the lead author of the co-authored work he or she wrote the text

Name: *Ali Atharifard* Signature:  Date: *18.10.2017*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit.

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Publication:

L. Vanden Broeke, A. Atharifard, B. P. Goulter, J. L. Healy, M. Ramyar, R. K. Panta, M. Anjomrouz, M. Shamshad, A. Largeau, K. Mueller, M. F. Walsh, R. Aamir, D. J. Smithies, R. Doesburg, K. Rajendran, N. J. A. de Ruiter, D. Knight, A. Chernoglazov, **H. Mandalika**, C. J. Batemana, S. T. Bell, A. P. H. Butler, P. H. Butler, *Oblique fluorescence in a MARS scanner with a CdTe-Medipix3RX, Journal of Instrumentation*, Vol. 11, Num. 12.

Please detail the nature and extent (%) of contribution by the candidate:

For this paper, I was a consultant for knowledge transfer. I participated in the discussions on the implication of energy response calibration using oblique XRF on the quality of visualisation. It is expected to improve the quality of Visualization. Contribution – 5%

Certification by Co-authors:

If there is more than one co-author then a single co-author can sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the candidate was the lead author of the co-authored work he or she wrote the text

Name: *Ali Atharifard* Signature:  Date: *18.10.2017*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit.

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Submitted:

Muhammad Shamshad, Marzieh Anjomrouz, Derek J Smithies, Antoine Largeau, Gray Lu, Ali Atharifard, Lieza Vanden Broeke, Raja Aamir, Raj Kumar Panta, Michael F Walsh, Brian P Goulter, Kishore Rajendran, Srinidhi Bheesette, Joe L Healy, Niels de Ruiter, Alex Chernoglazov, **Harish Mandalika**, Robert Doesburg, Stephen T Bell, Christopher J Bateman, Anthony P Butler, and Philip H Butler, *Semi-Analytic X-ray Source Model for MARS Spectral CT*, Submitted to IEEE Transactions on Medical Imaging (TMI) Journal.

Please detail the nature and extent (%) of contribution by the candidate:

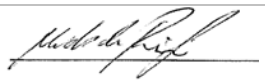
For this paper, I was a consultant for knowledge transfer. I participated in the discussions on the implication of the source model on the quality of visualisation. This will improve the quality of MARS data visualisation and thereby improve the quality of diagnosis. Contribution – 2%

Certification by Co-authors:

If there is more than one co-author then a single co-author can sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the candidate was the lead author of the co-authored work he or she wrote the text

Name: *Niels de Ruiter* Signature:  Date: *24-Oct-2017*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit.

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

To be submitted:

C. Bateman, D. Knight, B. Brandwacht, J. Mc Mahon, J. Healy, R. Panta, R. Aamir, K. Rajendran, M. Moghiseh, M. Ramyar, D. Rundle, J. Bennett, N. de Ruiter, D. Smithies, S. Bell, R. Doesburg, A. Chernoglazov, **H. Mandalika**, M. Walsh, M. Shamshad, M. Anjomrouz, A. Atharifard, L. Vanden Broeke, S. Bheesette, N. Anderson, S. Gieseg, T. Woodfield, P. Renaud, A. Butler, P. Butler, MARS-MD: rejection based image domain material decomposition, Submitted to Journal of Instrumentation (JINST).

Please detail the nature and extent (%) of contribution by the candidate:

For this paper, I was a consultant for knowledge transfer. I participated in the discussions on the implication of the two algorithms of MARS-MD on the visualisation of the resultant material datasets, with focus on the visualisation artefacts from misclassified material voxels and consequently the validity of the volume images. Contribution – 3%

Certification by Co-authors:

If there is more than one co-author then a single co-author can sign on behalf of all
The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the candidate was the lead author of the co-authored work he or she wrote the text

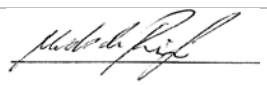
Name: *Niels de Ruiter* Signature:  Date: *24-Oct-2017*

Table of Contents

List of Figures

List of Tables

Acknowledgments

First of all, I am heartily thankful to my supervisors Dr. Christoph Bartneck, Dr. Anthony Butler, Dr. Philip Butler, Dr. Mark Billinghurst, and Dr. Niels de Ruiter, whose encouragement, supervision and continuous support throughout my Ph.D. enabled me to develop a deep understanding of the subject and complete my thesis successfully.

My sincere thanks to senior radiologist Dr. Michael Hurrell, my colleague Dr. Alexander Chernoglazov, the Christchurch Hospital, and the University of Otago (Christchurch) for making my evaluation study possible.

I am very grateful to Dr. Philip Butler, Dr. Anthony Butler and the MARS project for funding my research. I would like to thank the entire MARS team for their support. Last but never the least, thanks to all my colleagues at the HIT Lab for providing a great environment for my research.

Academic Contributions

The work reported in this thesis has contributed to several journal articles and a hybrid user interface (implemented in the MARS Vision software) that was later commercialized. This section lists the research and commercial outcome of my Ph.D.

Peer-reviewed journal articles

1. **Veera Bhadra Harish Mandalika**, Alexander I. Chernoglazov, Mark Billinghurst, Christoph Bartneck, Michael A. Hurrell, Niels de Ruiter, Anthony P. H. Butler, Philip H. Butler, A Hybrid 2D/3D User Interface for Radiological Diagnosis, Published in the Journal of Digital Imaging (JDI).

   This paper presents a novel 2D/3D desktop virtual reality hybrid user interface for radiology that focuses on improving 3D manipulation required in some diagnostic tasks. The hybrid system combines a zSpace stereoscopic display with 2D displays, and mouse and keyboard input. The paper also presents an evaluation of the system against the existing 2D only interface with a user study that involved performing a scoliosis diagnosis task. There were two groups of participants: medical students, and radiology residents. The results show that the hybrid interface is both more efficient, and more accurate than the traditional 2D only interface.

   For this paper, I developed a hybrid 2D/3D user interface in MARS Vision software for medical data exploration. I later explored various tasks to evaluate my interface and chose to use the scoliosis diagnosis task. I also designed and carried out a user study to evaluate my hybrid interface against the existing 2D radiology software. I chose medical students and radiology residents as two groups to compare the effects between novice and experienced groups of users for my study. I analysed the results and presented them along with my hybrid interface concept in the paper.

2. R. Aamir, A. Chernoglazov, C. J. Bateman, A. P. H. Butler, P. H. Butler, N. G. Anderson, S. T. Bell, R. Panta, J. L. Healy, J. L. Mohr, K. Rajendran, M. F. Walsh, N. J. A. de Ruiter, S. P. Gieseg, T. Woodfield, P. F. Renaud, L. Brooke, S. Abdul-Majid, M. Clyne, R. Glendenning, P. J. Bones, M. Billinghurst, C. Bartneck, **H. Mandalika**, R. Grasset, N. Schleich, N. Scott, S. J. Nik, A. Opie, T. Janmale, D. N. Tang, D. Kim, R. M. Doesburg, R. Zainon, J. P. Ronaldson, N. J. Cook, D. J. Smithies, K. Hodge (2014), MARS spectral molecular imaging of lamb tissue: data collection and image analysis, Journal of Instrumentation, Vol. 9, Num. 2.

   This publication described the use of the MARS molecular imaging system for lamb soft tissue imaging. The raw, pre-processed, and reconstructed data is available at: `http://hdl.handle.net/10092/8531`.

   For this publication, I helped develop the software MARS Vision, which was used to produce images to present the distribution of materials in a piece of lamb meat scanned in the MARS scanner. My work primarily consisted of implementing viewpoint camera manipulation.

3. K. Rajendran, M. F. Walsh, N. J. A. de Ruiter, A. I. Chernoglazov, R. K. Panta, A. P. H. Butler, P. H. Butler, S. T. Bell, N. G. Anderson, T. B. F. Woodfield, S. J. Tredinnick, J. L. Healy, C. J. Bateman, R. Aamir, R. M. N. Doesburg, P. F. Renaud, S. P. Gieseg, D. J. Smithies, J. L. Mohr, **V. B. H. Mandalika**, A. M. T. Opie, N. J. Cook, J. P. Ronaldson, S. J. Nik, A. Atharifard, M. Clyne, P. J. Bones, C. Bartneck, R. Grasset, N. Schleich, M. Billinghurst. (2014), Reducing beam hardening effects and metal artefacts in spectral CT using Medipix3RX, Journal of Instrumentation, Vol. 9, Num. 3.

   This is the second of two publications to present spectral CT data from the MARS molecular imaging system to the wider scientific community. The paper itself presents a validation of reducing beam hardening effects through narrow energy bins. Along side the paper, the raw, normalized, reconstructed and material decomposition datasets were uploaded for public access at `http://hdl.handle.net/10092/8851`.

   For this publication, I helped develop the software MARS Vision, which was used to produce images for 3D visualisation. These images demonstrated the

reduction in beam hardening effects through narrow energy bins which is possible using the MARS scanner. My work consisted of implementing effective viewpoint and light position manipulation controls.

4. A. Atharifard, J. L. Healy, B. P. Goulter, M. Ramyar, L. Vanden Broeke, M. F. Walsh, C. C. Onyema, R. K. Panta, R. Aamir, D. J. Smithies, R. Doesburg, M. Anjomrouz, M. Shamshad, S. Bheesettea, K. Rajendran, N. J. A. de Ruiter, D. Knight, A. Chernoglazov, **H. Mandalika**, S. T. Bell, C. J. Bateman, A. P. H. Butler, P. H. Butler, Per-pixel energy calibration of photon counting detectors, Journal of Instrumentation, Vol. 12, Num. 3.

   This paper presents a technique for per-pixel energy calibration of photon-counting x-ray detectors (PCXDs) that quantifies the energy response of individual pixels relative to the average response. This technique takes advantage of the measurements made by an equalized chip. It uses a known global energy map to quantify the effect of threshold dispersion on the energy response of the detector pixels across an energy range of interest. The additional information provided by this per-pixel calibration technique can be used to improve spectral reconstruction.

   For this paper, I was a consultant for knowledge transfer. I participated in the discussions on the implication of per-pixel energy calibration on the quality of visualisation. It is expected that this technique will be integrated into the MARS imaging and will improve the quality of image processing and thereby improve the quality of MARS data visualisation.

5. L. Vanden Broeke, A. Atharifard, B. P. Goulter, J. L. Healy, M. Ramyar, R. K. Panta, M. Anjomrouz, M. Shamshad, A. Largeau, K. Mueller, M. F. Walsh, R. Aamir, D. J. Smithies, R. Doesburg, K. Rajendran, N. J. A. de Ruiter, D. Knight, A. Chernoglazov, **H. Mandalika**, C. J. Batemana, S. T. Bell, A. P. H. Butler, P. H. Butler, Oblique fluorescence in a MARS scanner with a CdTe-Medipix3RX, Journal of Instrumentation, Vol. 11, Num. 12.

   This paper aims to improve the spectral performance of the MARS imaging chain by improving the calibration of the energy response of the detector. A common method for x-ray calibration is to use x-ray fluorescence (XRF). The

oblique (off-axis) XRF in the MARS scanner is measured, when a mono-atomic foil is placed in front of the x-ray source. The issue is to identify the optimal position that maximizes the XRF. A theoretical model is presented to identify the location in the detector plane to obtain maximum XRF. Preliminary measurements are taken and compared with the model. The results show that the model helps determine an optimal XRF field, but further refining of the model is needed.

For this paper, I was a consultant for knowledge transfer. I participated in the discussions on the implication of energy response calibration using oblique XRF on the quality of visualisation. It is expected to improve the quality of Visualization.

Articles submitted for peer review

1. Muhammad Shamshad, Marzieh Anjomrouz, Derek J Smithies, Antoine Largeau, Gray Lu, Ali Atharifard, Lieza Vanden Broeke, Raja Aamir, Raj Kumar Panta, Michael F Walsh, Brian P Goulter, Kishore Rajendran, Srinidhi Bheesette, Joe L Healy, Niels de Ruiter, Alex Chernoglazov, **Harish Mandalika**, Robert Doesburg, Stephen T Bell, Christopher J Bateman, Anthony P Butler, and Philip H Butler, Semi-Analytic X-ray Source Model for MARS Spectral CT, Submitted to IEEE Transactions on Medical Imaging (TMI) Journal.

   This paper presents the development of a parametrized semi-analytical x-ray source model is in the diagnostic imaging range (30-120 kVp) by applying the regression techniques to data obtained from custom Monte Carlo simulations of the x-ray tube used in numerous MARS scanners. Additionally, a comparison of the model with experimental data collected with a MARS scanner is also presented.

   For this paper, I was a consultant for knowledge transfer. I participated in the discussions on the implication of the source model on the quality of visualisation. It is expected that in future, this model can be used to improve quantitative accuracy of reconstruction resulting is higher quality attenuation volume, improving visualisation and diagnosis.

To be submitted for peer review

2. C. Bateman, D. Knight, B. Brandwacht, J. Mc Mahon, J. Healy, R. Panta, R. Aamir, K. Rajendran, M. Moghiseh, M. Ramyar, D. Rundle, J. Bennett, N. de Ruiter, D. Smithies, S. Bell, R. Doesburg, A. Chernoglazov, **H. Mandalika**, M. Walsh, M. Shamshad, M. Anjomrouz, A. Atharifard, L. Vanden Broeke, S. Bheesette, N. Anderson, S. Gieseg, T. Woodfield, P. Renaud, A. Butler, P. Butler, MARS-MD: rejection based image domain material decomposition, Submitted to Journal of Instrumentation (JINST).

   This paper outlines image domain material decomposition algorithms that have been routinely used in MARS spectral CT systems.These algorithms (known collectively as MARSMD) are based on a pragmatic heuristic for solving the under-determined problem where there are more materials than energy bins.Two algorithms based on this process are presented. Firstly the Segmentation variant, which uses segmented material classes to define each sub-problem; and secondly the Angular Rejection variant, which defines the rejection criteria using the angle between reconstructed attenuation vectors. Several examples are given of use of the two MARS-MD algorithms for 6-material decomposition.

   For this paper, I was a consultant for knowledge transfer. I participated in the discussions on the implication of the two algorithms of MARS-MD on the visualisation of the resultant material datasets, with focus on the visualisation artefacts from misclassified material voxels and consequently the validity of the volume images.

Users of my interface

My hybrid interface is integrated into the MARS Vision software, which is a commercial product that is currently being sold. This software is currently the only viewer that supports visualisation of MARS material images.

My interface has become an integral part of the MARS System. A custom visualisation workstation including the zSpace stereoscopic display and NVIDIA's Quadro graphics card is included with the commercial distribution of the MARS system.

Currently my interface as a part of the MARS System is being used for pre-clinical research at the following institutions.

1. University of Notre Dame

   https://www.nd.edu/

2. Mayo Clinic

   http://www.mayoclinic.org/

3. Joint Institute for Nuclear Research (JINR)

   http://www.jinr.ru/main-en/

4. University of Canterbury

   http://www.canterbury.ac.nz/

5. University of Otago, Christchurch

   http://www.otago.ac.nz/christchurch/index.html

6. Rensselear Polytechnic Institute

   http://rpi.edu/

In addition, MARS Vision is also being used at the following institutions, as a standalone system (not as a part of the MARS System).

1. Luxbright

   http://www.luxbright.com/

   A copy of MARS Vision software along with visualisation hardware was purchased by Luxbright, specifically for using my hybrid interface to explore and visualize their own data.

2. HIT Lab NZ

   http://www.hitlabnz.org/

   My interface is being used to study and demonstrate human computer interaction in fish-tank virtual reality based interfaces.

3. Lincoln University

   http://www.lincoln.ac.nz/

   Only MARS Vision software was used for pre-clinical research that involved exploring regions of interest within a scan of sheep brain using regular CT.

Chapter I

Introduction

This Ph.D. thesis details the work conducted to develop interactive tools for the exploration of spectral computed tomography (CT) datasets produced by the MARS scanner for medical diagnosis/imaging. In particular, the focus is to improve the 3D manipulation of volumetric data. The key outcome from this research is the design of a novel 2D/3D hybrid user interface for interactive exploration. A study showing the effectiveness of the hybrid interface in comparison with the standard 2D interface for experienced as well as novice users in diagnostic radiology is presented.

Medical imaging refers to a number of non-invasive techniques that are used to look inside the body. This, in turn, can be used for the diagnosis or planning treatment of several medical conditions. For example, most surgical and cancer treatments rely on it [1]. It has been considered one of the most valuable medical developments in the past 1,000 years by the 'New England Journal of Medicine' [2]. Popular examples of medical imaging include radiography, magnetic resonance imaging, ultrasound, tactile imaging, photoacoustic imaging, echocardiography, and tomography.

One of the emerging medical imaging modalities is spectral CT [3,4,5]. Unlike conventional CT that acquires a dataset over the wide range of the x-ray spectrum, spectral CT is capable of acquiring multiple CT datasets over differing energy ranges. The selected energy ranges are analogous to the different colours of visible light. This analogy for visible light can be used to distinguish conventional CT (greyscale), dual energy CT (two colours), and spectral CT (multiple colours). An illustration of this can be seen in figure 1.1.

The MARS project [6] is at the forefront of spectral CT imaging. The MARS small bore scanner is the world's first spectral CT scanner designed for pre-clinical research that is commercially available for research organizations across the world.

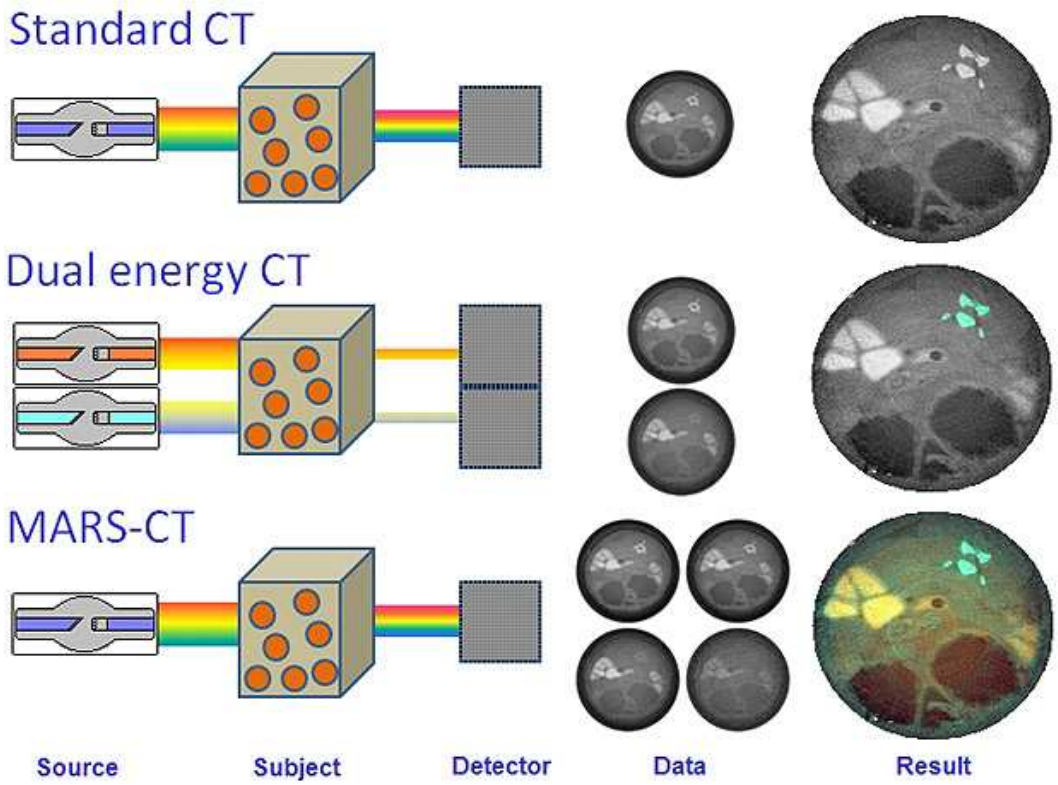


Figure 1.1: Difference between standard CT (top), dual-energy CT (middle) and spectral CT(bottom). Image courtesy of the MARS project.

By providing high-resolution imagery, the MARS scanner aims to advance the diagnosis and treatment of major health diseases such as atherosclerosis [7] and fatty liver disease [8,9]. The key feature of this system lies in the capacity to derive the densities (mg/ml) of targeted materials from the x-ray colours. [10, 11].

Spectral CT has the potential to achieve higher image quality and quantitatively classify materials for the same radiation dose. The MARS material data is similar to multi-modal medical imaging data, where multiple materials from the same volume represent their respective densities. It is also unique because each material, while being highly specific, lacks a complete context. For example, a calcium channel would show all bones but no soft tissues. This motivates the need for new volumetric visualization techniques.

The MARS project is aiming to develop a human-scale spectral CT scanner for medical diagnosis. There are many specialized human diagnosis tasks that rely on 3D manipulation such as measuring brain aneurysms, lung nodules, scoliosis

angles, etc. Effective 3D manipulation can result in a better diagnosis for such tasks. Current pre-clinical MARS researchers perform tasks such as measuring atherosclerosis, studying bone implants, etc. As these research topics mature into diagnostic protocols, they would also require some level of 3D manipulation.

The scanner produces spectral CT datasets that are visualized by the tool: MARS Vision. MARS Vision was developed specifically for visualizing MARS spectral datasets as no other tool was available at the time. The work presented in this thesis has resulted in the design of a novel hybrid interface that has been incorporated into MARS Vision. The interface has since become an integral part of MARS Vision and is currently being used by all MARS users for exploration of MARS spectral datasets at various sites around the world.

The first half of the thesis focuses on the interactivity improvements to the interface of MARS Vision. An overview of the tool as at the start of my Ph.D. is provided. A set of requirements are laid out based on the interactivity limitations of MARS Vision. These were derived from both MARS user feedback and empirical evidence from other studies in the literature. Various improvements were made to the tool that directly address these limitations. Additionally, some features were added to facilitate further research. These improvements and features include:

- Improving camera based manipulation by implementing the two-axis valuator interaction method. Modifying the algorithm to prevent the loss of rotational degrees of freedom (DOF) and preserving the object position during 3D rotation when using the camera panning or the clipping planes (section 3.1.1).

- Improving 2D slice manipulation by adding the flexibility for the user to map the mouse interactions to various 2D slice actions (section 3.1.2).

- Extending the MARS MCRT based volume rendering algorithm to support stereoscopic 3D rendering to provide binocular depth cues as well as facilitate future research with 3D interfaces (section 3.3).

- Implementation of mesh extraction to facilitate the extraction of 3D models from MARS spectral datasets to facilitate development of real-time inter-

faces (section 3.4). The meshes are currently being used by MARS users for exploration. They are also an integral part of the hybrid interface.

- Development of a real-time rendering engine to facilitate further research (section 3.5). This engine was later used to design the 3D GUI for the zSpace interface as well as the 2D/3D hybrid user interface.

- Implementation of the arbitrary slice to enable exploration of an oblique plane (section 3.6). The arbitrary slice was later used in a diagnostic radiology task for evaluating the hybrid interface.

The long term goal of the MARS project is to scale up the current scanner and commercialize it for human diagnostic imaging. This will be similar to the current form factor of radiology machines that are based on single energy CT. The quality of diagnosis depends on the radiologist's ability to identify features or anomalies within the data and accurately measure and report the findings. Some diagnosis tasks involve the rotation of one or more of the three anatomical planes. These tasks are often difficult using conventional radiology software, as they involve using a 2D input device such as a mouse to manipulate a 2D plane in 3D. This is also the case for exploring of MARS data using MARS Vision. Mouse input is precise for 2D manipulation tasks; however, previous research shows that using the mouse for 3D manipulation can be difficult [12, 13]. It is challenging to design a system that improves 3D manipulation while maintaining the benefits of mouse interaction.

The second half of the thesis focuses on the exploration of 3D input devices for more effective 3D manipulation. These are discussed along with the implementation of SpaceMouse and zSpace stylus input for MARS Vision. The motivation for a hybrid user interface is discussed along with the interface design. The methodology for evaluating the hybrid interface in the field of diagnostic radiology is described.

Finally, a statistical analysis of the results from the study showing the effectiveness of the hybrid interface is presented and discussed. The study compared the hybrid interface with a standard 2D interface using a scoliosis diagnosis task involving novice and experienced user groups. The analysis showed that the hybrid interface was more efficient for the novice group and more accurate for both groups when compared to the standard 2D interface.

In conclusion, this research has:

- Explored the interactivity limitations of MARS Vision as at the start of my Ph.D. with emphasis on 3D manipulation and formulated a set of requirements to address the limitations and facilitate further research.

- Implemented an effective mouse-based 3D manipulation method (the two-axis valuator [14]) based on the comparison in literature. Modified the algorithm to solve some issues caused by the type of 3D manipulation and other tools in MARS Vision.

- Implemented various features to facilitate further research including the stereoscopic 3D rendering, the mesh extraction, the arbitrary slice and a real-time rendering engine.

- Explored literature for 3D input devices and implemented the SpaceMouse and the zSpace interface. Designed a novel 2D/3D hybrid user interface combining the traditional 2D workflow and the zSpace interface.

- Evaluated the hybrid interface against the 2D interface in a diagnostic radiology scenario for experienced and novice users. The evaluation proved the effectiveness of the hybrid interface over the 2D interface.

- Integrated the interface into MARS Vision to make it available to all MARS users as part of the MARS product. The interface can be used in conjunction with all the other visualisation tools to explore MARS spectral datasets.

1.1 Thesis Structure

The thesis is structured into six chapters. An outline of the thesis structure along with a brief description of each chapter is given below.

Chapter 2 provides an overview of the MARS imaging toolchain with emphasis on the visualisation tool: MARS Vision. It gives an overview of the tool as at the start of my thesis, highlighting its various interactivity limitations and establishing a guiding principle for the research direction of this thesis.

Chapter 3 describes the interaction improvements that I made to MARS Vision based on the requirements established in Chapter 2. It also describes the features that I implemented in MARS Vision to facilitate further research such as the stereoscopic 3D rendering, the rapid mesh extraction, the arbitrary slice, and the rendering engine.

Chapter 4 describes various 3D input devices I explored along with my implementation of the SpaceMouse and the zSpace stylus input for MARS Vision. It also describes my design of the novel 2D/3D hybrid user interface and hybrid interaction along with my motivation behind the interface design.

Chapter 5 covers the evaluation of the hybrid interface. It starts by exploring the radiology workflow. It presents the evaluation methodology describing the participants, the task, the process, the measures (performance and usability), and the data analysis strategy. It also provides a statistical analysis of the results.

Chapter 6 concludes the thesis by wrapping up the results from the evaluation followed by an in-depth discussion of the findings, its impact, and where this should lead to in the future.

Chapter II

Requirements for interactive exploration

In this chapter, the existing MARS imaging toolchain is described with emphasis on the visualisation software: MARS Vision. Some of the existing features of MARS Vision (as at the start of my Ph.D.) are discussed such as data loading, 2D visualisation, 3D visualisation (e.g. volume rendering), viewpoint camera manipulation, and other tools. Additionally, the performance and interaction issues with MARS Vision are highlighted. Some of these issues are directly addressed in Chapter 3. These issues also help lay out the requirements for interactive exploration of MARS spectral data and establish a guiding principle for the research direction in Chapter 4.

The raw data acquired by the MARS scanner undergoes a significant amount of pre-processing before it is ready for visualisation and exploration. The set of applications that are responsible for the pre-processing of raw data, as well as the visualisation tool, are collectively called the MARS imaging toolchain. The toolchain consists of a set of applications and scripts that are responsible for tasks such as acquiring projection data during a scan, denoising of projection images, reconstruction of projection images into attenuation volumes, extraction of material volumes from the reconstructed attenuation volumes, and visualisation of both attenuation and material volumes. Section 2.1 explains the MARS imaging toolchain in detail.

The final part of the MARS imaging toolchain consists of the visualisation tool: MARS Vision. This tool was developed specifically for visualizing MARS data. It provides the means to explore MARS material data in 2D and 3D, and extract important information from specific regions of interest. The tool is currently intended for use in biological, scientific, and pre-clinical applications. Section 2.2 provides an overview of MARS Vision prior to my work.

Section 2.3 outlines the limitations of MARS Vision at the beginning of my Ph.D. project. These include performance, design, and usability limitations. Finally, a set of requirements for interactive exploration of MARS data are laid out in section 2.4.

2.1 MARS imaging toolchain

The MARS project is developing a complete spectral imaging system, which consists of the MARS scanner hardware (see figure 2.1) as well as tools for processing and analysing the data acquired by it. These tools are collectively called the MARS imaging toolchain. This section discusses the current structure and functions of the MARS imaging toolchain, with a focus on the flow of data from acquisition to visualisation. Interactive visualisation, the primary topic of this thesis is implemented within the software: MARS Vision. This is also the final part of the toolchain.

Figure 2.1: The MARS small bore spectral CT scanner.

The toolchain enables the smooth transition of data acquired by the MARS scanner hardware, to a form suitable for analysis. It has undergone various changes over the past few years. Only the current version of the toolchain (as of the writing

of this thesis) will be described (see figure 2.2). For more information regarding older versions of the toolchain and its evolution to the current stage, refer to the Ph.D. thesis by Alexander Chernoglazov [15].

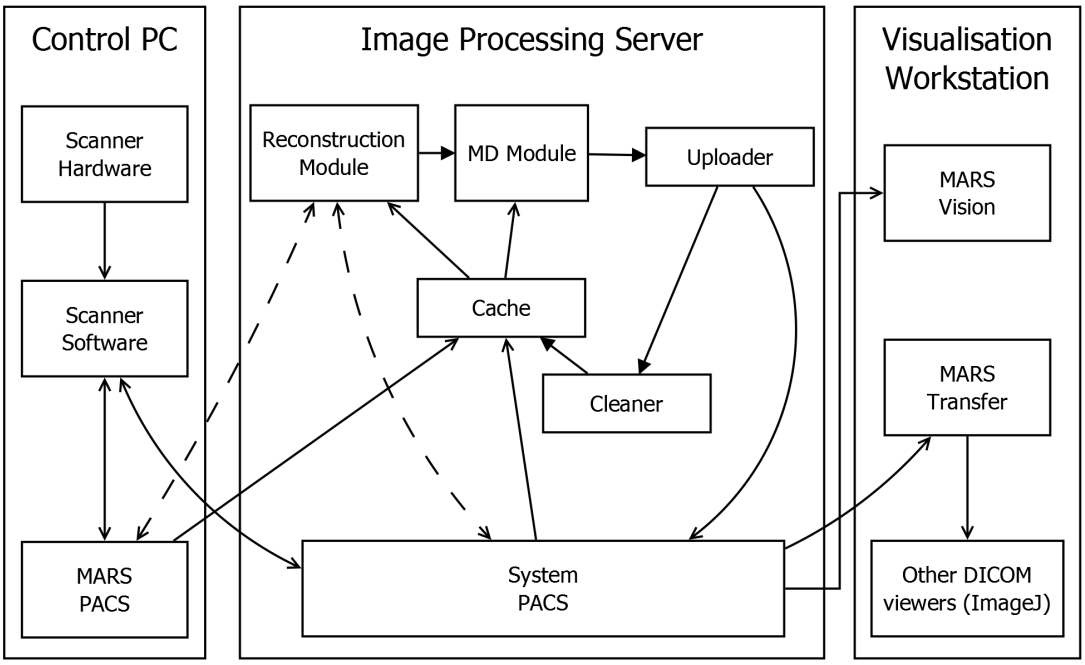


Figure 2.2: The current structure of MARS imaging toolchain in late 2017. The arrows indicate the possible flow of data. The filled arrows in Image processing server indicate the flow of actions in image processing. The dashed lines represent an alternate flow of data which is optional.

The toolchain is spread across three main physical components in the MARS system. These are the control PC, the image processing server, and the visualisation workstation. The control PC is always located on the MARS scanner. It is responsible for controlling the MARS scanner hardware and the image acquisition. Both the image processing server, as well as the visualisation workstation, can be physically located anywhere on the local network. The former is responsible for reconstruction, and material decomposition (MD), while the latter is used for analysing the material data.

The flow of data in the MARS imaging toolchain is as follows:

1. The scanner hardware acquires the raw projection images from a scan. These images are sent to the scanner software module.

2. The scanner software forwards them to the MARS picture archiving and communication system (PACS) [16] in the case of a calibration scan, or to the System PACS in all other cases.

3. When a reconstruction is requested by the user, the projection data from one of the PACS is accessed by the reconstruction module and stored in the cache. After reconstruction, the attenuation volumes are stored in the cache.

4. These attenuation volumes are then accessed by the MD module for material classification [17]. This process estimates the density of a set of target materials from the attenuation volumes. For example, a scan of a piece of meat can be decomposed into material volumes representing the densities of lipid, calcium (bone), and water (muscle). Once completed, the material volumes are sent to the uploader.

5. The Uploader takes the attenuation volumes and the material volumes and uploads them back to the PACS.

6. The attenuation and material volumes may then be accessed for visualisation either through MARS Vision, which can directly visualise the data or through MARS Transfer, which prepares the data for a 3rd party tool such as ImageJ [18].

2.1.1 PACS and data formats

PACS

There are two PACS units in the MARS system, namely, the MARS PACS and the system PACS. The MARS PACS is located on the control PC, while the system PACS is usually located on the image processing server. Research institutions that have a pre-existing PACS have the option to forward the data from system PACS to their own PACS.

The MARS PACS is used to store data from calibration scans, such as calibration phantoms. Calibration phantoms consist of vials of materials with known concentrations, which are used for material classification. This data can be accessed by the scanner software to configure the hardware for a scan. All data

stored on the MARS PACS at any site is accessible to the MARS team. This may be used for diagnosing problems at a site.

The system PACS is used to store data from all scans done at the site. MARS team can only access data on the system PACS if the site permits them to do so. The reconstructed volumes, as well as material data, is also stored on the system PACS.

Data formats

There are three main forms of data. These are the projection images, the attenuation volumes, and the material volumes. The projection images are obtained from the scanner. These are then used to reconstruct slices of images for each energy range. The set of images obtained after reconstruction are called attenuation volumes.

Standard CT uses Hounsfield units (HU) (see Equation 2.1) to quantify CT data, which is obtained from a linear transformation of attenuation coefficients based on the arbitrary definitions of air(-1000 HU) and water (0 HU) [19].

$$HU = \frac{\mu - \mu_{water}}{\mu_{water} - \mu_{air}} \times 1000 \tag{2.1}$$

For MARS datasets, linear attenuation coefficients are used for attenuation volumes instead of the standard Hounsfield units. This is because the attenuation volumes are intermediary datasets that finally result in material volumes. The Hounsfield units are energy-dependent [20] and the attenuation volumes have multiple energy ranges. Finally, these volumes are processed to classify materials in terms of density (mg/ml), thus resulting in material volumes.

Currently, all data is stored using the Digital Imaging and Communications in Medicine (DICOM) format. The DICOM format is a standard for storing and transferring medical images from several imaging systems (such as CT scanners) to various PACS systems [21]. The format is divided into several parts, where the storage and file exchange is described in *Part 10* [22]. The DICOM files contain image data as well as a set of tags that are used to store meta-data about the image.

The projection data uses private tags to record all scan specific information. The reconstructed data is based on the standard CT DICOM format with some

minor alterations. These alterations primarily include the support for multi-frame images using the standard DICOM tags and a more generic rescale to support linear attenuation instead of Hounsfield units. In the near future, the DICOM committee is expected to release a standardized multi-energy CT format that will be used when it is available. All MARS DICOM data may be viewed using standard DICOM viewers.

2.2 MARS Vision

MARS Vision is the software used for visualizing and analysing the MARS attenuation and material volumes. It is a very important part of the toolchain because it is the only tool that provides specific features for exploring MARS material density volumes for analysis. Therefore it is important that this tool is well designed, easy to learn, and use. It is also crucial that this tool provides sufficient features to enable the users to extract important information from MARS images. The 2D user interface for MARS Vision can is shown in figure 2.3.

The application is primarily composed of components that include dataset loading, 2D visualisation, 3D visualisation, viewpoint camera and lighting manipulation, and other tools. A comprehensive description of MARS Vision can be found in the Ph.D. thesis by Alexander Chernoglazov [15] (mainly in Chapters 5-8).

2.2.1 Data loading

MARS Vision can load datasets stored in two different formats: a raw binary format and the DICOM format. The former was a legacy format used by the MARS team, while the latter is the standard medical image format currently being used in the MARS data processing toolchain.

The raw binary format consisted of a raw file, that contained all the voxel data related to a single scan. This included all the attenuation volume data or all the material data for each scan. The layout of the data, along with other configuration tags were stored in an accompanying text-based descriptor file. Since the shift to using the DICOM standard, this format is not used any more. However, the format is still supported to facilitate access to old datasets.

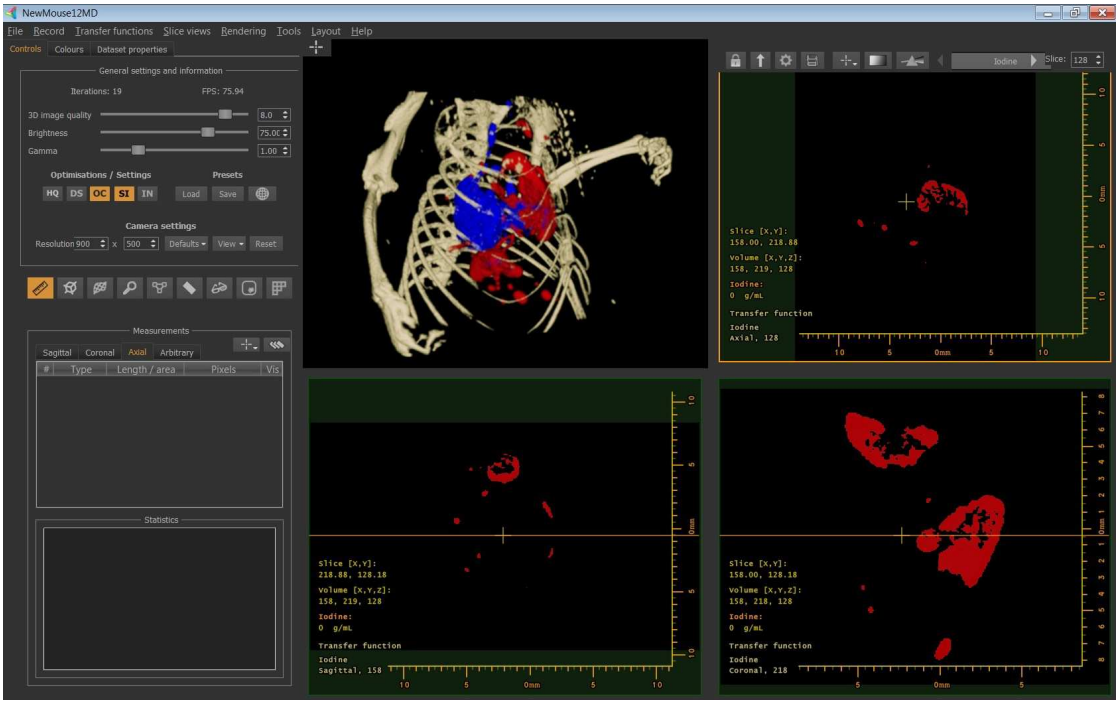


Figure 2.3: The 2D interface of MARS Vision showing the MARS Mouse12 dataset [10]. Notice the 3D view (top left), the axial slice view (top right), the sagittal view (bottom left) and coronal view (bottom right).

The DICOM format is currently being used for all MARS datasets. Both attenuation volumes, as well as the material data, is stored as DICOM images. The datasets are stored on the PACS and can be retrieved for analysis, through MARS Vision.

2.2.2 2D Visualization

MARS Vision supports 2D visualisation of volume data in the form of a slice view. Originally, it only supported three standard slice view orientations: sagittal, axial, and coronal. Later an arbitrary slice view was added to show an oblique slicing plane. The functionality of each view, including the user interface elements, was almost identical.

The slice views can be classified into two categories: axis-aligned, and arbitrary. Each axis-aligned slice view shows a slice, that lies in one of the three anatomical planes (coronal, sagittal, and axial). The arbitrary slice view shows a slice that

can be rotated to the desired orientation. This view was added to MARS Vision during my Ph.D., in order to facilitate interactive exploration. The arbitrary view is also integrated with the 3D view as discussed in section 3.6 and its integration into the hybrid interface is discussed in section 4.4.2.

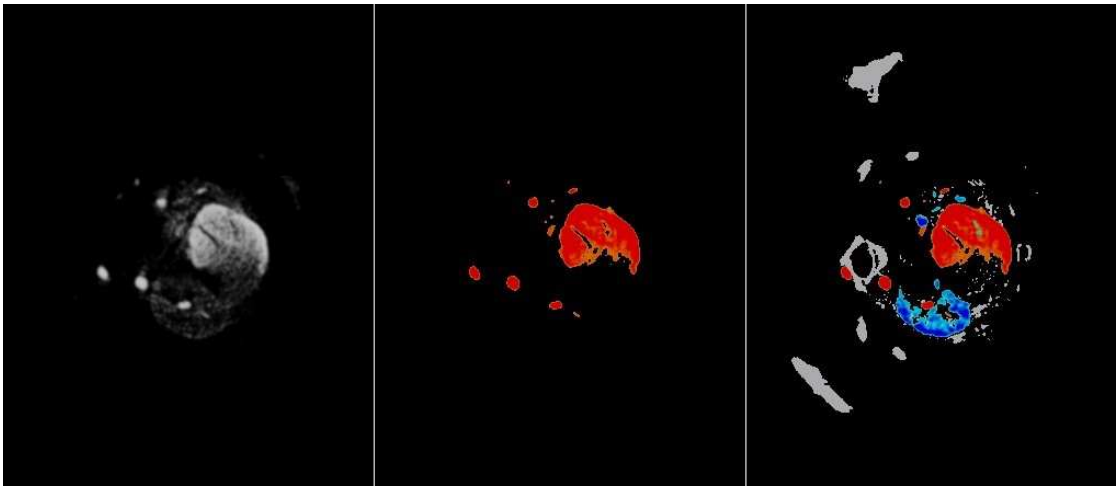
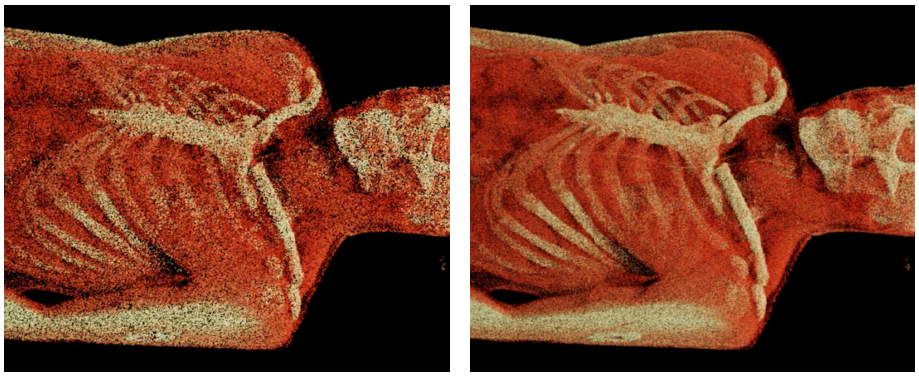


Figure 2.4: The Mouse12 dataset [10] visualized with MARS Vision, showing the axial slice view, with the three display modes: window and level (left), transfer function (middle), and spectral mode (right). The iodine channel was used for the first two modes and the spectral mode shows all materials with different transfer functions.

The slice views offer the standard slice manipulation functionality, such as scrolling, zooming, and panning. They also allow the user to change the window and level settings [23] to control the level of information displayed within the slice. Additionally, they also offer an option to replicate the effects of the transfer function used for volume visualisation (see section 2.2.3). Finally, they support a visualisation technique called "spectral mode", which blends the slices from multiple MARS energy or material volumes together. These modes can be seen in figure 2.4.

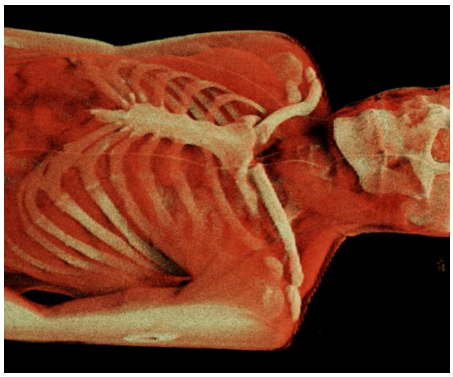
2.2.3 Volume rendering

MARS Vision was initially built on Exposure Render, which implemented a single-volume Monte-Carlo ray tracing (MCRT) DVR algorithm for volume rendering [24]. This algorithm works by iteratively refining the 3D image over multiple steps. This can be seen clearly in figure 2.5.

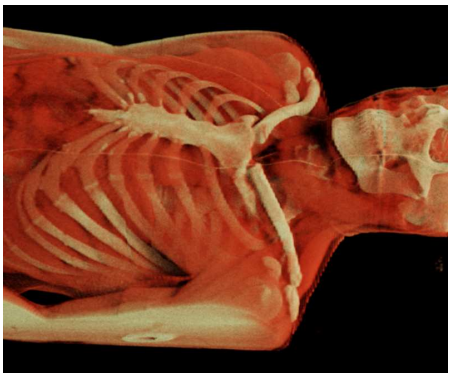


(a) Iteration 1

(b) Iteration 5



(c) Iteration 20

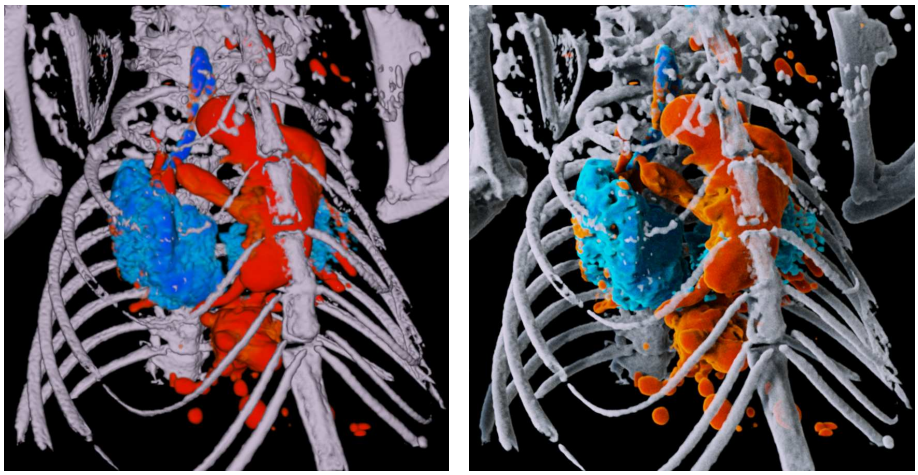


(d) Iteration 80

Figure 2.5: 3D images of the Visible Human Male dataset [25] at various iterations of the MCRT DVR algorithm.

Each *iteration* consists of five steps. The first step is stochastic raycasting, where rays are cast from the camera position towards the volume and each ray finds a scattering point inside the volume, resulting in a frame estimate. The second step is blurring of the frame estimate using a Gaussian kernel. The third step is Monte-Carlo integration of frame estimate where all prior frame estimates are accumulated. The fourth step is tone mapping, where colour correction and brightness adjustments are performed. Finally, the fifth step is denoising (using the K-nearest neighbour filter [26]).

Currently, a modified version of this algorithm by Alexander Chernoglazov [15], that supports rendering of multiple material volumes is used, see figure 2.6b. I will refer to this algorithm as the MARS MCRT volume rendering algorithm for the remainder of this thesis. Alternatively, a ray marching DVR algorithm (an



(a) Using ray marching (b) Using MCRT DVR algorithm

Figure 2.6: 3D images of multiple material volumes from the Mouse12 MARS dataset using MARS Vision.

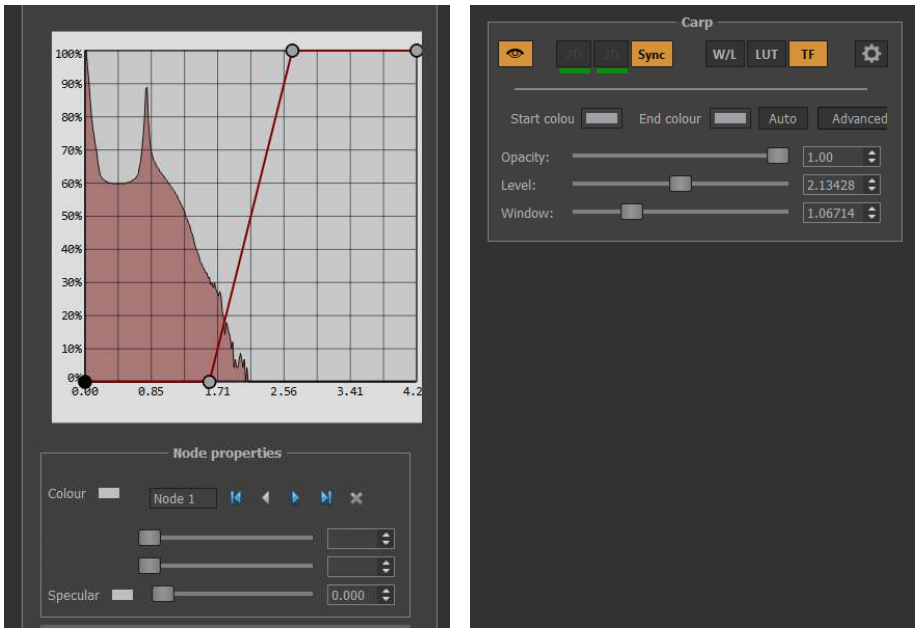
extended version of the algorithm proposed by Kajiya and Von Herzen [27]) is also supported, see figure 2.6a. The former is primarily used for generating 3D images for presentations and publications as it produces a more photo-realistic image, while the latter is usually used for inspection.

Both volume rendering algorithms are implemented using compute unified device architecture (CUDA) [28]. The simple ray marching algorithm was later modified to support minimum and maximum intensity projections referred to as MinIP and MIP respectively.

MARS Vision also supports a transfer function editor to control the information displayed in the 3D image, see figure 2.7a. A slider-based interface was developed specifically for working with MARS material volumes, see figure 2.7b. The modified MCRT algorithm, as well as the other features, were developed by Alexander Chernoglazov during his Ph.D. [15].

2.2.4 Camera and light manipulation

The 3D volume rendering in MARS Vision uses three vectors for the viewpoint camera. These are the *position*, *target*, and *up* vectors. The position vector defines the viewpoint camera position. The target vector defines the position of the centre of the 3D volume. The up vector defines the up direction of the camera



(a) Standard transfer function editor. (b) Slider-based transfer function editor.

Figure 2.7: Transfer function editor interface for DVR in MARS Vision.

that matches the top of the rendered image.

These vectors are then used to obtain the *direction* vector ($\vec{direction} = target - source$). This *direction* is then normalized and used to obtain the *left* vector ($\vec{left} = cross(\vec{direction}, \vec{up})$). Both these vectors along with the *up* vector are used for 3D volume rendering. This is the same procedure used in the ray marching DVR algorithm as well as the MCRT-based algorithm in MARS Vision.

By default, the camera is set to view the dataset from the *Front* camera orientation, which is the coronal view. MARS Vision provides tools to snap the camera orientation to *Back*, *Top*, *Bottom*, *Left*, and *Right* views. Additionally, the camera can also snap to isometric combinations such as *FrontLeftTop*, *FrontRightTop*, etc.

The light sources make use of a 360° latitude and longitude approach to defining their relative position to the centre of the volume. The interface consisted of sliders to control the distance, latitude, and longitude of each light source, see figure 2.8.

These values are then converted to two angles: theta (azimuth) and phi (elevation). These angles along with the specified distance are used to compute the

Figure 2.8: The MARS Vision UI showing the controls to manipulate the 3D light source.

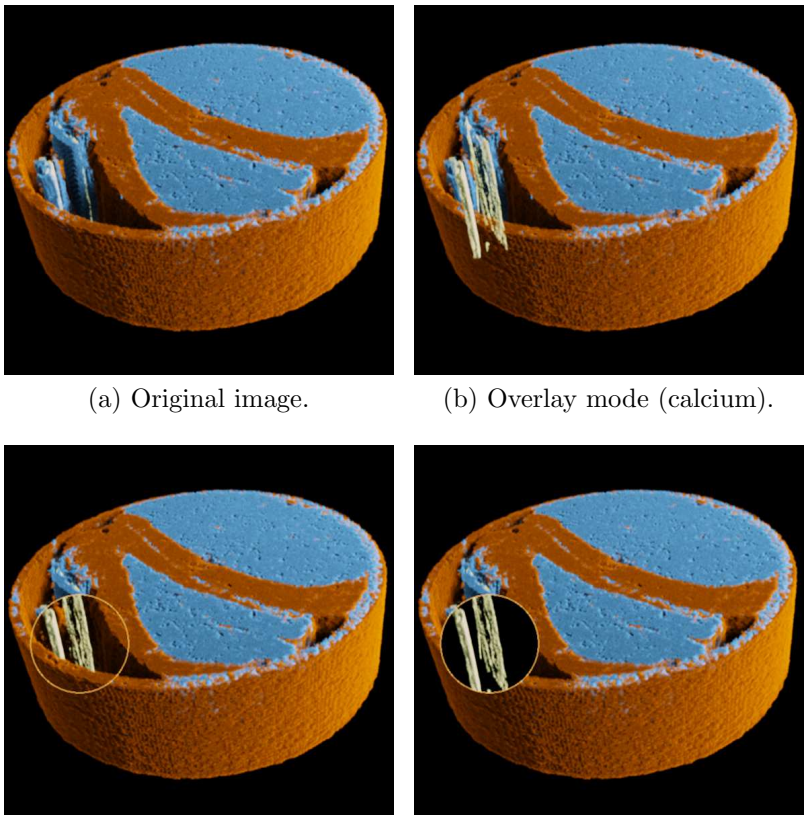
light's *position*. The *target* for all lights is the centre of the volume. Hence, this is used in conjunction with the position of each light source to obtain the direction for each point light.

2.2.5 Other tools

MARS Vision has various other tools for exploring MARS datasets. These tools were developed to provide the means to explore and visualize information from MARS attenuation and material volumes. Some tools focused on quantification, while others focused on simplifying the 3D visualisation. Some of the tools were developed as a response to requests from the MARS Vision users (pre-clinical researchers from the MARS team and the University of Otago), while others were inspired by other commercial tools for exploring standard CT data.

There are two tools for reducing occlusion of various materials in 3D volume rendering. Firstly, the overlay mode allowed a certain material to be overlaid over other materials, see figure 2.9b. This allowed the overlaid material to be clearly visible without occlusion from other materials. Secondly, the magic lens tool allowed selected materials to be hidden within a defined region, see figures 2.9c and 2.9d. This could be used to hide materials that occlude the regions of interest and observe the underlying materials. This tool is also available for use with the 2D views.

MARS Vision also has several measurement and annotation tools similar to



(a) Original image. (b) Overlay mode (calcium).

(c) Magic lens (calcium, lipid). (d) Magic lens (calcium).

Figure 2.9: The MARS meat dataset is used to show the MARS Vision tools to reduce occlusion of the calcium material.

commercial DICOM viewers. These include the line, rectangle, polygon, and ellipse measurements tools, as shown in figure 2.10. These tools provide information for a selected shape on a slice. The measurement tools can also provide statistical information for the entire slice.

Although these tools are limited to the 2D slice views in MARS Vision, they can be extended to 3D visualisations. A set of such tools and 3D interaction techniques for measuring distances, angles, and volumes are discussed by Preim et al [29]. The 3D line and angle measurement tools were also implemented during this thesis (see section 4.3.5) to facilitate the evaluation discussed in Chapter 5. The implementation of these tools for quantitative measurements in virtual environments is discussed in detail by Hagedorn et al [30].

The information relating to a dataset such as a camera orientation, light source

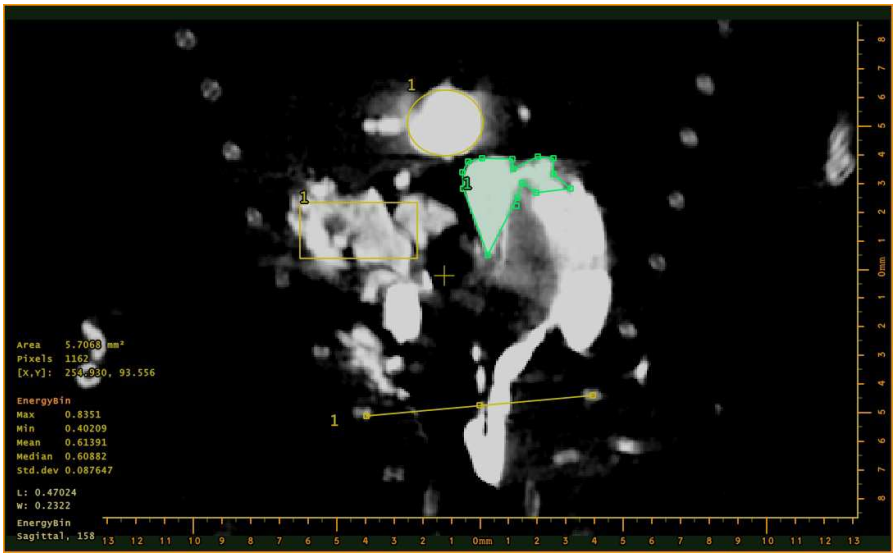
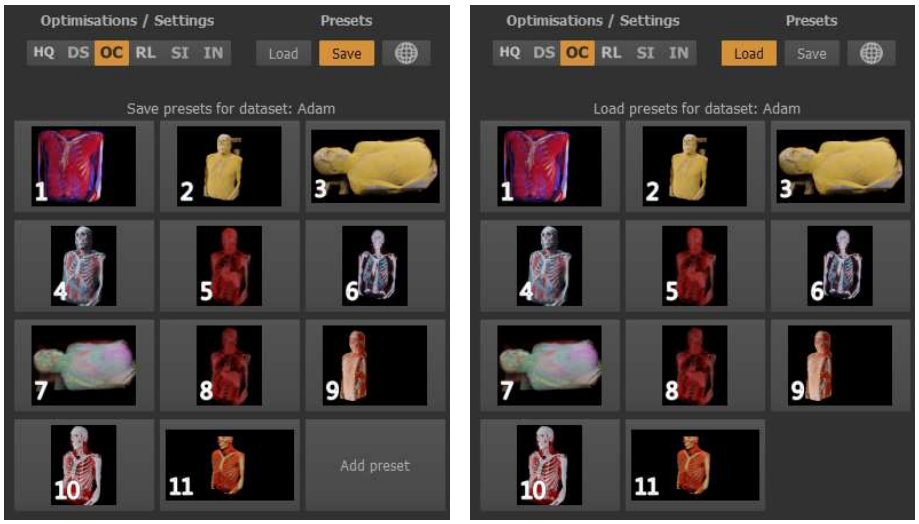


Figure 2.10: Slice measurement tools in MARS Vision showing the line, rectangle, ellipse and polygon measurements on a sagittal slice of the MARS Mouse12 dataset. The polygon measurement is currently selected and its statistics are displayed in the bottom left area of the image.

position, transfer function properties, measurements, and various other attributes related to exploring a dataset can be stored in files called presets. A preset is a saved state while exploring a dataset that can also be used to share information with other users. The interface for saving and loading presets can be seen in figure 2.11.

Additionally, MARS Vision also contains tools for editing and annotating on the surface of the 3D volumes. The voxel deletion tool allows the user to remove spherical chunks of voxels by selecting a point on the surface. The selection is performed with a mouse and the sphere radius is user-defined. The 3D point is chosen by projecting a ray from the camera through the mouse position until it intersects the 3D surface of the displayed 3D volume. Thus, the selection of a 3D point is limited to the visible surface of the 3D volume.

The 3D annotation tool allows the user to define 3D points on the surface of the volume and annotate them with text. The tool also allows defining multiple points, forming a convex hull [31]. This convex hull can then be assigned a text description. Finally, MARS Vision supports clipping planes along the x, y, and z axes. Clipping planes can be used to either exclude visualisation of unwanted



(a) Saving a preset. (b) Loading a preset.

Figure 2.11: The save and load controls UI in MARS Vision for presets of the visible human dataset. [25]

features or reveal hidden features within the dataset.

By default, the clipping planes are situated at the extreme ends of the dataset. The planes exclude all data on one side of the plane. Thus, by adjusting the clipping planes, the user is able to select a sub-region within the volume and exclude everything else from the visualisation. This can be used to remove unwanted features that lie near the edges of the dataset or to view a slice through the volume along with the contextual information from the rest of the volume.

2.3 Limitations of MARS Vision

MARS Vision was designed to visualize and measure the MARS attenuation and material datasets, which was not possible with standard DICOM viewers, as they do not support MARS data formats. Although the tool is capable of producing high-quality 3D volumetric images and displaying 2D slice information, there exist limitations in interactivity and performance. This section explores some of the limitations of MARS Vision that have been addressed in this thesis. There have been no formal usability studies to deduce the limitations. Therefore, this section is based on the known limitations at the time of the design, as well as the issues

reported by the MARS users.

2.3.1 Performance limitations

One of the major concerns from MARS Vision users is the performance of the 3D visualisation. While the MCRT based algorithm used for 3D visualisation produces images of relatively high quality, it is not possible to render these images at that quality every frame. The 3D image is built up over several frames before it achieves the desired quality. A typical image at a resolution of 1920×1080 with 60% quality and 200 iterations takes approximately 75 seconds. While the image is built up, the view is updated at approximately 2 frames per second (FPS). This slow rendering time prevents intuitive interaction with the graphics.

While interacting with the 3D model, the rendering quality is substantially reduced (approximately 10%) to allow the user to interact without freezes. However, in some cases, the lower quality 3D image does not provide sufficient detail and could distract the user from achieving the desired angle or position.

The MCRT algorithm used in MARS Vision was a modified version of the algorithm from Exposure Render [24]. This algorithm was aimed to build the image in iterations in order to allow some interactivity for the user. Usually, the volume image was built over 200 iterations. This value was chosen from crude observations and is not an optimal value. This is similar to the passive GPU raycasting approach used for graphics production [32]. While the graphics production approach assumes the viewpoint is set beforehand and builds the image by composing multiple lighting characteristics, the algorithm used in MARS Vision composes the required lighting for each iteration and then combines all the iterations incrementally. Thus after each iteration, the user sees a more complete and realistic representation of the final image that appears to refine itself.

However, a major difference between DICOM viewing and graphics production is that the DICOM viewing requires the viewpoint to change according to the user's commands. When the user interacts with the volume, the rendering needs to start from the first iteration each time any viewing parameters are changed. For example, if the user rotates the viewpoint, the volume image needs to restart building up from the first iteration.

Thus, for the whole period of the user rotating the viewpoint in a continuous

swipe gesture using the mouse, the volume remains at the first iteration, see figure 2.5a. This is also the case while performing any interaction such as zooming, rotating, panning, or manipulating the light. At higher image quality, this often leads to frame rates of less than 1 FPS which prevents users from performing interactive actions.

The performance of the MARS MCRT volume rendering algorithm can slow down substantially in cases where large datasets are visualized or high-quality settings are used. In such cases, the time taken to render a single iteration can reach several seconds. This results in the user not being able to interact consistently since the visual feedback is substantially delayed.

The MARS Vision user interface may become inactive due to the GPU being bottlenecked by the MARS MCRT volume rendering algorithm. Often this bottleneck also results in other background operating system user interface updates getting frozen. This can have a detrimental effect on the user experience.

The workaround was to severely reduce the rendering quality (by 50%) as well as the view resolution to interact with the volume. Once the desired transfer function, viewpoint orientation, as well as lighting position were achieved, the user would then increase the quality and resolution to get the final image. This was not only tedious but resembled the offline graphics rendering workflow more than an interactive real-time tool.

2.3.2 Design limitations

Although the DICOM standard is used to store the MARS datasets, not all standard DICOM tags are supported by MARS Vision. These include the patient orientation tags that might be set during a scan, to correct the orientation for standard human anatomical views. This results in the slice views being oriented incorrectly, or at times, swapped.

Another key limitation includes the inability to support arbitrarily scaled datasets. MARS datasets consist of voxels that are always adjacent and cubic. Hence MARS Vision was designed to only handle datasets with uniformly scaled voxel data. If the irregular voxel shapes and spacing are not considered, not only will the 3D visualisation of such datasets look incorrect, but any measurements made will also be incorrectly scaled. The scaling can be performed manually as a quick solution,

but this can become tedious for repeated tasks.

Due to these limitations, MARS Vision was not able to visualize some standard CT human datasets. Support for most of the commonly used standard CT DICOM tags as well as irregular voxel shapes and spacing will provide the following benefits:

- MARS Vision can load and visualize all standard CT datasets from all vendors.
- Support for all standard CT datasets will enable more rigorous testing of MARS Vision features.
- MARS Vision will have proper support and compliance with the DICOM standard.

2.3.3 Interactivity limitations

The traditional mouse is the primary input device for MARS Vision. The user interface makes use of familiar windows based UI elements such as tabs, menus, checkboxes, buttons, radio buttons, sliders, spinners, and so on. However, when it comes to interacting with the 3D volume using the mouse, the interaction was designed poorly. The initial implementation of two-axis mouse-based rotation had a major flaw. The rotation was based on used mutually independent Euler angles derived from the mouse motion in the two axes. The angle accumulated an error at every instance, and over time when the accumulated error exceeded 90°, this would result in loss of a degree of freedom. This phenomenon is called gimbal lock [33]. This made it difficult for the user to achieve the desired orientation.

The light source manipulation was performed using three parameters which were designed as sliders. The user needed to adjust the latitude, longitude and distance sliders to position the light source in a spherical orbit around the centre of the volume (see figure 2.8). Research shows that people find it difficult to relate sliders to rotations about separate axes [34].

The MARS Vision users also reported finding this interface very difficult to use. There were no visual cues to help the user determine the current location of the light source. The user had to rely on the changes in lighting on the volume, to mentally compute the light source's location. This was so difficult for users, that they avoided using the light source manipulation tool altogether.

2.4 Requirements for interactive exploration

This section attempts to summarize the requirements for interactive exploration. These requirements are based on the limitations of the existing tool, discussed in the previous section (2.3). Although the requirements are focused on the interaction with 3D volumes, there are a lot of technical requirements to facilitate the development of new tools and algorithms, for MARS Vision. The requirements are classified into two categories: *technical* and *usability*. The aim was to not only provide better tools for interaction to explore the 3D MARS information but also to ensure that these tools are more efficient than existing tools as well as are easy to learn and use.

2.4.1 Technical Requirements

The technical requirements for interactive exploration are based on the design of MARS Vision. MARS Vision is built using the Qt framework [35] for the graphical user interface (GUI) and the CUDA architecture [36] for the 3D volume rendering algorithm. An open source 3D visualization software called the visualization toolkit (VTK) [37], was also used to push the volume rendered frames to the 3D view within the Qt user interface. The user interface, as well as the volume rendering algorithm, were heavily modified to suit the needs of the MARS program.

However, the frameworks were not modified. Since there was no commonly used graphics API such as Direct3D [38] or OpenGL [39] and there was no rendering engine in place, development of new tools was very difficult. Thus the major technical requirement was the development of a rendering engine to facilitate future research based on MARS Vision. Other technical requirements were based on the limitations of the existing tool. These are listed in the table 2.1.

2.4.2 Usability Requirements

MARS Vision was built to visualize MARS datasets. MARS Vision includes many tools built to explore spectral information, however, none of these tools were formally evaluated for usability or efficiency. This was because the software was changing too rapidly and there was insufficient time to evaluate a specific version of the software.

Table 2.1: Technical Requirements

№	Requirement	Explanation
1	Improving camera manipulation	It is important to fix the existing interactivity problems with the camera manipulation before exploring new techniques as this can then be used for comparison with other techniques.
2	Developing a rendering engine	This would not only improve the efficiency of the existing rendering pipeline in MARS Vision but would also enable the rendering of various 3D models that can be used to develop 3D user interfaces for future research.
3	Exploring stereoscopic 3D	This will improve depth perception of the volume images in MARS Vision by providing binocular cues. This is particularly beneficial due to the existing difficulty in light source manipulation. This can also be used for future research with stereoscopic 3D interfaces such as those used in virtual reality (VR).
4	Rapid mesh extraction	Since the DVR algorithm in MARS Vision is unable to achieve interactive frame rates consistently, it is important to explore mesh rendering as an alternative. Meshes can easily be rendered consistently on higher resolutions displays with a relatively minor loss in performance. This will facilitate the research of 3D input devices that demand consistently higher frame rates. Additionally, the meshes can be exported and used for 3D printing.
6	Implementing the arbitrary slice	The arbitrary slice view is a specialized tool used to explore arbitrary slicing planes in radiology. This tool will not only provide similar functionality to MARS Vision but also provide a means to test new 3D manipulation techniques using existing tasks in radiology.

Although some modifications were made to MARS Vision as a result of reported usability issues, only a small number of issues were identified and fixed this way. Since this thesis aims to improve 3D manipulation for exploring 3D volumetric data, it is not enough to simply implement a better interaction design. It is required to evaluate the usability and effectiveness of the new interaction. Due to the lack of well-defined diagnosis protocols for MARS datasets, the only other possibility is to use standard CT data for evaluation. Since the exploration tasks are mostly similar, I expect that any benefits resulting from the improved 3D manipulation will also be similar. The usability requirements are listed in table 2.2

Table 2.2: Usability Requirements

№	Requirement	Explanation
1	Evaluation strategy	The key requirement is to find a task with standard CT that can be used to evaluate the interface. It is important to select the participants who are accessible and also represent a user group with similar image exploration requirements as the users of MARS Vision.
2	Design the user study	Once a strategy for evaluation is established, a user study must be designed and carried out.
3	Analysis of the results	The final step would be to analyse the results to assess the effectiveness of the new interaction.

2.5 Summary

In this chapter, I described the MARS imaging toolchain with an emphasis on MARS Vision. I discussed the limitations of MARS Vision, as it existed at the start of my thesis. I also described the interactivity limitations in detail. Finally, I derived a set of requirements for interactive exploration of MARS spectral data.

The key issue I wished to tackle in this thesis was to improve 3D manipulation for MARS volumetric data exploration. The requirements were divided into technical and usability categories. The technical requirements focus on improving

existing mouse-based 3D manipulation as well as the implementation of a rendering engine, which would facilitate implementation of a more advanced 3D user interface and input devices. The usability requirements focus on the development of an evaluation strategy. The key decision was to develop an evaluation strategy with standard CT data, due to the limited user base for MARS Vision.

In conclusion, it is important to improve the 3D manipulation available to users of MARS Vision. This will not only result in a more efficient and accurate exploration of MARS information but also make it easy for new users to learn and use the interaction tools.

Chapter III

MARS Vision : Preliminary work

In this chapter, I describe the preliminary work on MARS Vision. This work was necessary to facilitate the development of other tools and interfaces for research, as established in Chapter 2. The main focus of this chapter is on improving mouse-based manipulation and the rendering pipeline in MARS Vision.

In addition to these improvements, I also explore the extension of the MARS volume rendering algorithm for rendering in stereoscopic 3D, the implementation of rapid mesh extraction, and a real-time rendering engine to facilitate the development of 3D user interfaces for MARS Vision.

This chapter is structured as follows. Firstly, I explore the improvements to the mouse-based manipulation in section 3.1. The improvements to the rendering pipeline are detailed in section 3.2. The extension to the MARS volume rendering algorithm to generate stereoscopic 3D pairs is explained in section 3.3. The implementation of rapid mesh extraction is discussed in section 3.4. Here I also explore the computation of gradient normals for shading the extracted mesh, extraction of multiple meshes for MARS material volumes, rendering them in stereoscopic 3D, render cache optimization, and memory optimization for the mesh. Finally, the development of a real-time rendering engine for MARS Vision is described in section 3.5.

The modifications to camera-based manipulation described in section 3.1 and the improvements to the rendering pipeline discussed in section 3.2 were used to generate the 3D images that were published in two papers by Rajendran et al. ([40, 41]), for which I was a co-author.

The work in relation to stereoscopic rendering (see section 3.3), the rapid mesh extraction (see section 3.4), the real-time rendering engine (see section 3.5), and the arbitrary slice (see section 3.6) were used for implementing the hybrid interface discussed in Chapter 4. All the work from this chapter is implemented in the release

version of MARS Vision that is currently being used by MARS users at various locations.

3.1 Improving mouse-based manipulation

In this section, I explore the problems with mouse-based manipulation implemented in MARS Vision. I mainly focus on the problems with the mouse-based 3D camera manipulation. This is vital to the exploration of MARS volumetric images. I later explore the mouse-based 2D slice manipulation and various interaction designs. I implement a more flexible way to map the mouse-based interaction for slice manipulation. This flexibility in slice manipulation is later used to carry out the human evaluation, described in Chapter 5.

The user interaction with MARS Vision is primarily designed around the traditional mouse and keyboard input devices. While most of the GUI elements are similar to windows based applications, interacting with the slices and the 3D volumes is different. The user can choose to use GUI elements such as scroll bars, spinners, and buttons as well as the keyboard to interact with slice manipulation (see figure 3.1). The mouse can also be used to directly interact with the 2D slices or the 3D volume. While directly manipulating using the mouse is very efficient, there are many challenges.

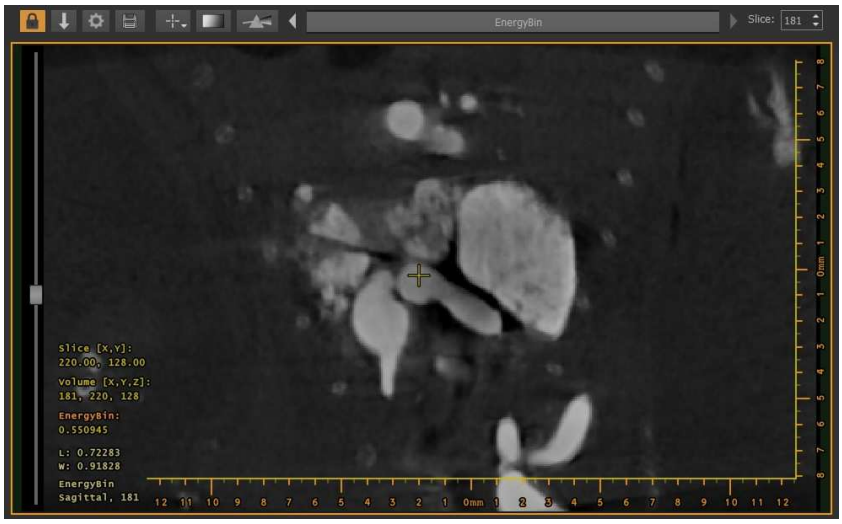


Figure 3.1: Slice view GUI for MARS Vision showing the slice scroll bar (left), various buttons (top) and a slice spinner (top right) for interacting with the slice.

3.1.1 Camera manipulation

Rotating a 3D volume with a 2D mouse is a complicated task as this involves the user providing 2D input on a plane that is then mapped to a 3D rotation. Hence, it is important to implement this correctly. There are many ways to map 2D mouse motion to 3D rotation.

The most commonly used 3D rotation techniques using the mouse input include Bell's virtual trackball (VT) [42], Shoemake's VT (called ArcBall) [43], two-axis valuator [14] and two-axis valuator with fixed up axis [44].

Four principles for designing 3D rotation using the mouse input were proposed by Bade et al. [45]. The first principle states that same actions must result in the same rotations. The second principle states that the direction of the 2D motion should match the direction of the 3D rotation. The third principle states that rotations must be transitive. The fourth principle states that the method should allow a control to display ratio. The control to display ratio is a factor that is used to scale the sensitivity of the 3D rotation.

Bell's VT was an improvement over the method from Chen et al. Chen's VT projected the moving mouse position to a virtual sphere behind the viewport and used the projected points on the sphere to compute the 3D rotation [14]. This often resulted in an axis of rotation that was not perpendicular to the motion of the mouse. This was improved by Bell's VT who projected the mouse motion on a viewport to a virtual sphere and a hyperbola. This provided a smoother interaction compared to Chen's VT.

However, Bell's VT was very dependent on the position of the mouse in the viewport. The direction of motion would change for horizontal mouse motion near the top and bottom edges of the viewport. Shoemake's ArcBall was a variation of Chen's VT where the angle of rotation was scaled by a chosen factor. This allowed the rotation to be transitive. The mathematics for all the virtual trackballs are described in detail by Henriksen et al [46].

Chen's two-axis valuator method mapped 2D mouse motion to angles about two axes (up and right) [14]. The horizontal mouse movement was mapped to rotation about the up vector and the vertical mouse movement was mapped to rotation about the right vector (the vector perpendicular to the up and view vectors). Later, a variation to this method where the up vector was fixed was used in the

commercial 3D modelling software 3DS MAX [44].

The usability comparison by Bade et al [45] found that the two-axis valuator with fixed up vector was the most efficient method followed by the simple two-axis valuator. Later, a comparison of Bell's VT, ArcBall, and two-axis valuator by Zhao et al. [47] found no significant difference in performance or accuracy between the methods. More recently an evaluation by Rybicki et al. [48] compared ArcBall and two-axis valuator and found no significant difference in performance (task completion time) between the two.

Research shows that there was no significant difference in the performance or accuracy between the VT and two-axis valuator methods. However, according to the 3D rotation design principles [45], two-axis valuator fulfilled three out of the four principles while other methods only met two. The two-axis valuator method also allowed to easily define a control to display ratio for rotation. Hence, I chose to use the two-axis valuator method [14] for camera rotation in MARS Vision. The implementation I used for the two-axis valuator method is shown in algorithm 1.

The *from* vector $\vec{F}$ represents the position of the camera and the *target* vector $\vec{T}$ represents the position the camera is looking at. The direction vector $\vec{D}$ is derived by $\vec{F} - \vec{T}$. The *up* vector $\vec{U}$ represents the up direction for the camera. These three vectors ($\vec{F}$, $\vec{D}$, and $\vec{U}$) are used to define the camera's position and orientation relative to the volume for rendering volumes in MARS Vision.

The $\vec{T}$ vector is set to the centre of the volume. The camera's position depends on the initial view mode selected by the user. The starting values for $\vec{F}$ position as well as the $\vec{U}$ axis vary based on the viewing mode. By default, the viewing mode is from the front. This can be changed to back, left, right, top or bottom. The $\vec{F}$ and $\vec{U}$ change accordingly to snap the camera to the respective position. The user is then able to rotate the camera from this starting position.

One of the key issues for this algorithm is the possibility of gimbal lock, which is the loss of one DOF for rotation. If the $\vec{U}$ vector is rotated to a direction that is parallel or opposite to the direction vector $\vec{D}$, their cross product will result in zero. Thus no further rotation is possible in that axis. While it may be hard to encounter, since both vectors $\vec{U}$ and $\vec{D}$ are rotated by the same angles about the same axes, the rotations often accumulate error due to floating point rounding errors. This error is very minor, but since this is accumulated every pass when the mouse is being dragged, the accumulated error adds up quickly.

Algorithm 1 The MARS Vision's 3D rotation function. This function is executed while the mouse is dragged with its left button pressed down.

Data: Mouse motion in the X and Y axis as Δ_x and Δ_y. The Camera's current from $\vec{F}$, target $\vec{T}$ and up $\vec{U}$ direction vectors.

Result: The rotated camera's position (from) $\vec{F}$ and up $\vec{U}$ vectors.

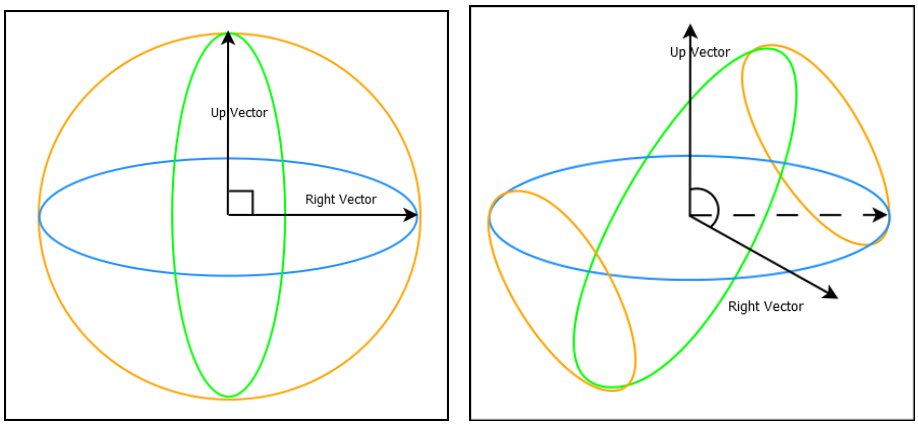
initialization:
Target to camera vector $\vec{D}$
Right axis vector $\vec{R}$
Azimuth angle $\angle A$
Elevation angle $\angle E$
Rotation scaling constant S

orbit the camera
if *Mouse left button pressed and mouse moving* **then**
  $\vec{D} \leftarrow Normalize(\vec{F} - \vec{T})$
  $\vec{R} \leftarrow Cross(\vec{D}, \vec{U})$
  Normalize vector $\vec{R}$
  $\angle A \leftarrow \Delta_x \times \pi \times S$
  $\angle E \leftarrow \Delta_y \times \pi \times S$
  rotate $\vec{D}$ by $\angle A$ around axis $\vec{U}$ and $\angle E$ around axis $\vec{R}$
  rotate $\vec{U}$ by $\angle A$ around axis $\vec{U}$ and $\angle E$ around axis $\vec{R}$
  $\vec{F} \leftarrow \vec{T} + \vec{D}$
end

The effect of the accumulated error results in the $\vec{U}$ and $\vec{R}$ axis vectors not being perpendicular to each other. This restricts the range of angles the camera can rotate around the volume as can be seen in figure 3.2. When the axis vectors are perpendicular, any orientation in the spherical region can be accessed by a combination of rotations around these axes.

However, when the axes are no longer perpendicular, only orientations within a hyperboloid region are accessible. As error accumulates, the axes deviate from being perpendicular and the accessible region reduces in volume. This continues until it collapses completely when the axes become parallel and the gimbal lock occurs.

When a gimbal lock occurs, only the up ($\vec{U}$) axis vector can be used for rotation in the algorithm 1. This is because the right axis vector ($\vec{R}$) becomes zero as the



(a) Perpendicular axes. (b) Non-perpendicular axes.

Figure 3.2: A figure demonstrating the $\vec{U}$ and $\vec{R}$ axis vectors used for rotation. The blue circle represents the rotation around the up vector, the green circle represents the rotation about the right vector and the yellow circle represents the spherical region in the left image and hyperboloid region in the right image.

cross product of parallel vectors is zero. The rotation in the algorithm about a zero-axis results in no rotation. Hence, the user will be unable to rotate about the $\vec{R}$ axis (*Azimuth*) any more.

Preventing gimbal lock

In order to solve this issue of partial or complete loss of rotational degrees of freedom, I implemented a modified algorithm. In order to negate the error accumulated from rotations, I introduced a step to adjust the up vector after each rotation to force it to be perpendicular to the direction vector. This would ensure that the resulting right vector remains perpendicular as well. This can be seen in algorithm 2.

The advantage of using this approach is that the axis vectors remain perpendicular, thereby preventing loss of rotational freedom. I achieve this by re-computing the up vector post rotation using the cross product of right axis vector and the direction vector. While this always forces the up vector to be adjusted relative to the direction, this might also induce a slightly inaccurate final rotation. This is due to the difference between the user perceived up vector and the adjusted up vector. But this is negligible and is too small to be noticed by the user.

Algorithm 2 MARS Vision's modified 3D rotation function. This function is executed while the mouse is dragged with its left button is pressed down.

Data: Mouse motion in the X and Y axis as Δ_x and Δ_y. The Camera's from $\vec{F}$, target $\vec{T}$ and up $\vec{U}$ direction vectors.
Result: Rotated camera's position (from) $\vec{F}$ and up $\vec{U}$ vectors.

initialization:
Target to camera vector $\vec{D}$
Right axis vector $\vec{R}$
Adjusted Up axis vector $\vec{\mu}$
Azimuth angle $\angle A$
Elevation angle $\angle E$
Rotation scaling constant S

orbit the camera
if *Mouse left button pressed and mouse moving* **then**
  $\vec{D} \leftarrow \vec{F} - \vec{T}$
  $\vec{R} \leftarrow Cross(\vec{D}, \vec{U})$
  Normalize vector $\vec{R}$
  $\angle A \leftarrow \Delta_x \times \pi \times S$
  $\angle E \leftarrow \Delta_y \times \pi \times S$
  rotate $\vec{D}$ by $\angle A$ around axis $\vec{U}$ and $\angle E$ around axis $\vec{R}$
  rotate $\vec{U}$ by $\angle A$ around axis $\vec{U}$ and $\angle E$ around axis $\vec{R}$
  $\vec{F} \leftarrow \vec{T} + \vec{D}$
  $\vec{\mu} \leftarrow Cross(\vec{R}, \vec{D})$
  $\vec{U} \leftarrow Normalize(\vec{\mu})$

end

Accumulating rotations in quaternions

This solution works most of the time and is very efficient at preventing partial or complete loss of rotational DOF. However, while it is not possible to build up the rotational error, it is still possible to perform a single pass with a rotational angle greater than 90°. This can cause the axes to lose perpendicularity in a single pass, leading to loss of rotational DOF that cannot be recovered by algorithm 2.

Thus, accumulating rotations directly in vectors is not an ideal solution. To avoid loss of axes perpendicularity and rotational DOF, rotations must be accu-

mulated separately. An efficient way to accumulate complex rotations is quaternions [49]. They can store arcs of rotations and have properties that allow the easy accumulation of various rotations.

So in order to solve the issue of gimbal lock and eliminate the possibility of loss of rotational DOF, I chose to store rotations in quaternions similar to the approach by Hanson et al [50]. I stored the initial *from*, *target*, and *up* vectors based on the initial viewing orientations selected. Then, all rotations performed by the user were accumulated in a quaternion. It also allowed a simple revert to the default orientation, by resetting the quaternion. Another advantage was that since I stored the rotation in a quaternion, re-computation of the up axis vector was no longer needed to preserve perpendicularity. This can be seen in algorithm 3.

The quaternions are used to store the accumulated rotational angle. This is done by converting the mouse motion to angles, which are then converted to quaternions $\mathbb{Q}_a$ (azimuth) and $\mathbb{Q}_e$ (elevation). Later these quaternions are multiplied with the accumulated rotation quaternion $\mathbb{Q}$ to accumulate the current rotation. While theoretically quaternion multiplication does not affect the normality of the quaternion, in practice, floating point precision issues can cause errors. These errors cause the quaternion to slowly deviate from normality causing similar axis faults to the first method. Thus to prevent this, the quaternion is normalized after each rotation.

The initial direction vector $\vec{I}$ and initial up vector $\vec{\mu}$ are rotated using the accumulated quaternion $\mathbb{Q}$ to compute the current camera's from $\vec{F}$ and up $\vec{U}$ vectors. One issue is that the distance of the camera from the target is lost since the rotation only accounts for the direction. Hence, this distance is stored in L and is later multiplied by the rotated direction vector.

Panning and rotation

Panning the camera is another frequently used camera manipulation. The user can explore various parts of the image quickly by panning the camera. Panning the camera results in moving the camera position and target in a 2D plane, which is perpendicular to the camera's direction. While this is a simple translation of the camera's *from* and *target* vectors in a 2D plane, this has a major effect on the camera's rotation.

Algorithm 3 MARS Vision's 3D modified rotation function with quaternion accumulation. This function is executed while the mouse is dragged with its left button is pressed down.

Data: Mouse motion in the X and Y axis as Δ_x and Δ_y. The camera's from $\vec{F}$, target $\vec{T}$ and up $\vec{U}$ direction vectors. The camera's initial direction vector $\vec{I}$ and intial Up vector $\vec{\mu}$. The rotation quaternion $\mathbb{Q}$.

Result: Rotated camera's position (from) $\vec{F}$ and up $\vec{U}$ vectors.

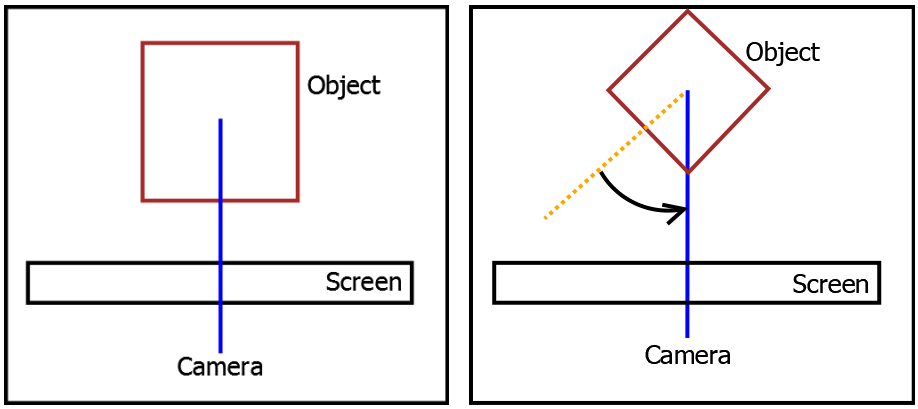
initialization:
Target to camera vector $\vec{D}$
Right axis vector $\vec{R}$
Direction vector length L
Adjusted Up axis vector $\vec{\mu}$
Azimuth $\angle A$
Elevation $\angle E$
Rotation scaling constant S
orbit the camera
if *Mouse left button pressed and mouse moving* **then**
  $\vec{D} \leftarrow \vec{F} - \vec{T}$
  $\vec{R} \leftarrow Cross(\vec{D}, \vec{U})$
  Normalize vector $\vec{R}$
  $L \leftarrow$ length of $\vec{D}$
  initialize quaternion $\mathbb{Q}_a$ from $\angle A$ and $\vec{U}$
  initialize quaternion $\mathbb{Q}_e$ from $\angle E$ and $\vec{R}$
  $\mathbb{Q} \leftarrow \mathbb{Q} \times (\mathbb{Q}_a \times \mathbb{Q}_e)$
  Normalize $\mathbb{Q}$
  $\vec{D} \leftarrow$ rotate $\vec{I}$ by quaternion $\mathbb{Q}$
  $\vec{D} \leftarrow L \times \vec{D}$
  $\vec{U} \leftarrow$ rotate $\vec{\mu}$ by quaternion $\mathbb{Q}$
  $\vec{F} \leftarrow \vec{T} + \vec{D}$

end

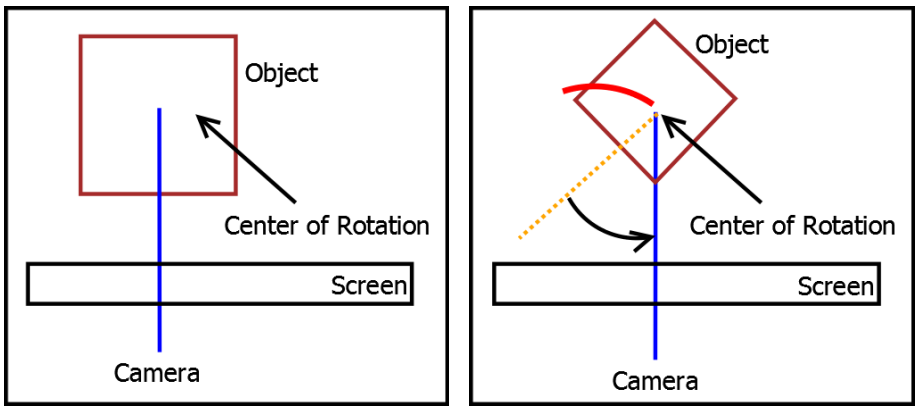
The camera orbits around the volume in a spherical region. This makes the model appear to rotate about its own centre from the user's perspective. But when the user pans the camera, the origin for rotation no longer lines up with the centre of the screen. Hence subsequent rotations appear to move the volume from its position.

Translations and rotations are not commutative. Thus panning the camera

changes the centre of rotation, thereby changing the way the object rotates on the screen. As seen in figure 3.3, the object appears to move in an arc when the camera is panned before performing the rotation. This can heavily hinder the user's ability to achieve the desired camera orientation for exploring the volume.



(a) Camera before rotation (no pan) (b) Camera after rotation (no pan)



(c) Camera before rotation (with pan) (d) Camera after rotation (with pan)

Figure 3.3: Figure showing how panning affects the camera rotation. Notice how the object remains at the centre of the screen in (a) and (b), while the object has moved in an arc (shown in red) on the screen when the camera is rotated after a pan in (c) and (d).

To solve this issue, the panning was separated from the camera's vectors. The camera pan was stored and accumulated in a 2D vector. This vector was used in conjunction with the camera's *from* and *target* vectors as input for the MARS volume rendering algorithm. This can be seen in algorithm 4.

The pan is accumulated by converting the mouse motion (Δ_x and Δ_y) into

Algorithm 4 MARS Vision's modified camera panning function with pan accumulation. This function is executed while the mouse is dragged with its middle button is pressed down.

Data: Mouse motion in the X and Y axis as Δ_x and Δ_y. The 3D view's window width as W and Height as H. The camera's from $\vec{F}$, target $\vec{T}$ and up $\vec{U}$ vectors. The camera's pan as 2D vector $\vec{P}$.

Result: Accumulated camera pan $\vec{P}$.

initialization:
Direction vector D
Right axis vector R
Direction vector length L
Horizontal pan length U
Vertical pan length V

pan the camera
if *Mouse middle button pressed and mouse moving* **then**
  $L \leftarrow$ length of $(\vec{F} - \vec{T})$
  $U \leftarrow L \times (\Delta_x / W)$
  $V \leftarrow L \times (\Delta_y / H)$
  $\vec{P} \leftarrow \vec{P} + Vector(U, V)$
end

using the pan before every iteration
$\vec{D} \leftarrow \vec{F} - \vec{T}$
$\vec{R} \leftarrow Cross(\vec{D}, \vec{U})$
Normalize vector $\vec{R}$
$\vec{F} \leftarrow \vec{F} + (\vec{P}_u \times \vec{R}) + (\vec{P}_v \times \vec{U})$
$\vec{T} \leftarrow \vec{T} + (\vec{P}_u \times \vec{R}) + (\vec{P}_v \times \vec{U})$

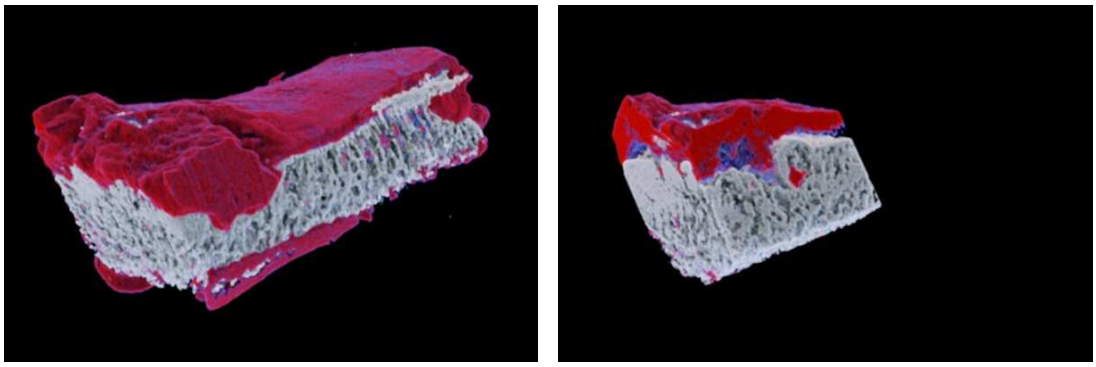
lengths (U and V) in the plane, perpendicular to the camera's direction. These are accumulated in a 2D vector $\vec{P}$. Before each volume rendering iteration, this pan is applied to the camera's from and target vectors. Since the pan only moves the vectors in a perpendicular plane the up axis vector remains unaffected.

Clipping planes and rotation

In MARS Vision, six axis-aligned clipping planes are provided that are located at the edges of the volume. These can be moved to restrict the amount of information

visualized from the volume. The data outside these edges is clipped out from rendering. They are often used to remove unwanted regions from the edges of the volume as well as to select a subset of the volume for visualisation.

When the volume is clipped, the visible portion of the volume is a subset of the entire volume and the centre of the volume no longer coincides with the centre of the clipped volume (see figure 3.4). Thus, this affects the camera rotation. The problem is that when the user rotates a clipped volume, its position is not preserved on the screen because it rotates about the centre of the entire volume.



(a) 3D Volume image (no clipping) (b) 3D Volume image (clipped)

Figure 3.4: Volume image from the MARS bone cartilage dataset showing the entire volume (a) and a clipped portion using clipping planes (b).

This problem is similar to how the panning affected the camera rotation, shown in figure 3.3. The clipped volume appears to move in an arc when rotations are performed. This motion is directly proportional to the volume that is clipped by the planes.

To solve this issue, instead of using volume's centre ($\vec{T}$), the clipped volume's centre is temporarily used as the target vector ($\vec{T_c}$). Thus, the camera's direction vector $\vec{D}$ to the clipped centre can be computed, which is later used to compute the right axis vector $\vec{R}$. This affects the rotation quaternions ($\mathbb{Q}_a$ and $\mathbb{Q}_e$) from algorithm 3. Later, the volume's centre $\vec{T}$ is used as the target vector for the volume rendering algorithm. This is similar to moving the clipped volume to the origin, rotating around it, and moving it back to its position.

This allows the user to rotate the clipped volume about its own centre. This also prevents forcing of unwanted translations to the volume, while clipping. After

performing the rotation, the user can choose to change the clipping planes to view the entire volume and its position is preserved on the screen.

3.1.2 Slice Manipulation

2D Slice manipulation is an important part of exploring MARS datasets, which also relates directly to clinical practice for diagnosing human datasets. The slices in the three axes are displayed in three anatomical planes: sagittal, coronal, and axial. These slices can be manipulated using GUI elements such as spinners and sliders. The slice manipulation is also mapped to the mouse input to make it easier and faster for the user to explore the slices.

The slice manipulation consists of five major interactions. These include scrolling, panning, zooming, changing the window/level, and contextual actions. The scrolling is used to cycle through slices in an orthogonal view. The zooming can be used to change the scale of the slice contents displayed on the screen using bi-linear interpolation. This is used to closely observe features within a slice. The panning can be used to move the slice contents within the view. The window/level settings are used to control the grey level that is displayed within the slice. The contextual actions depend on the current mode of the slice view, which include placing annotations and measurements.

These interactions are mapped to the fixed actions involving mouse buttons as well as the keyboard. By default, the slice panning is mapped to left mouse button, the window/level adjustment is mapped to the middle mouse button, the slice scroll is mapped to the scroll wheel, the slice zoom is mapped to the scroll wheel with the 'shift' key held down, and the contextual actions are mapped to the right mouse button.

This default setting is based on a commercial radiology DICOM software called Inteleviewer [51]. Inteleviewer also allows the user to re-map the mouse actions to any interaction. Hence, I added this re-map functionality to MARS Vision. The possible mappings are shown in figure 3.5.

This change is beneficial to the MARS users because it will allow experienced users to re-map the functionality to match their preference. This change was also required to avoid bias in the user study (see Chapter 5) involving experienced users who had significant prior experience with Inteleviewer.

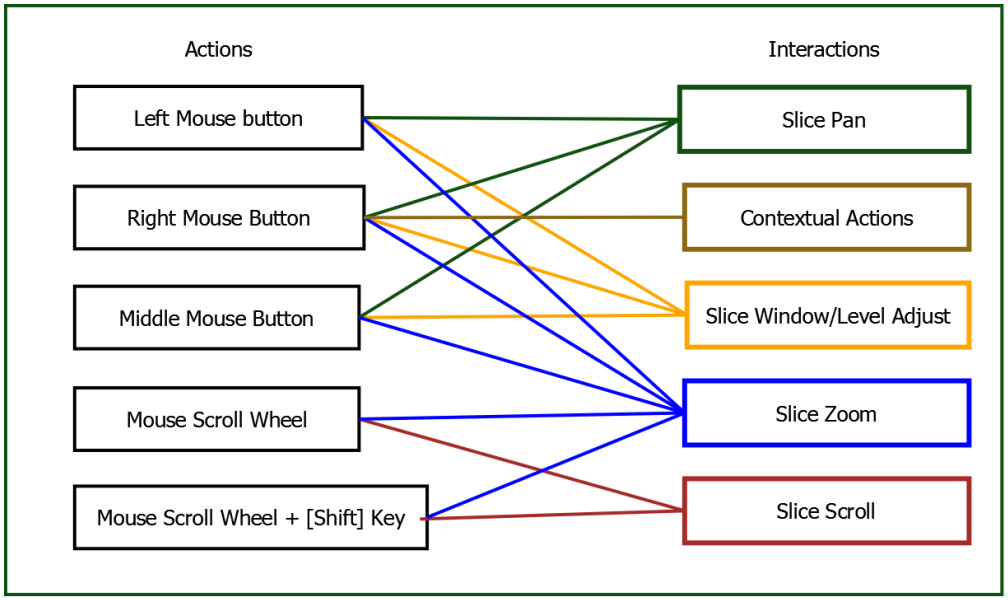


Figure 3.5: The possible mappings between the actions (left) and slice interactions (right) in MARS Vision.

3.2 Improving the rendering pipeline

The rendering pipeline in MARS Vision has a direct impact on the performance of the volume rendering. Since the MARS MCRT volume rendering algorithm progressively builds up the image, the minimum time taken to render the final image is also relatively high (at least a few seconds). Thus, it is difficult to notice the performance loss caused by the design of the rendering pipeline (few milliseconds).

At the start of my thesis, only the MARS MCRT algorithm for volume rendering existed in MARS Vision. MARS datasets were loaded from disk and stored on the system (host) memory. The data was then transferred to the graphics card's video (device) memory. Once it was transferred to the device memory, each frame was rendered using CUDA kernels and the final frame was copied back to the host memory. This frame was then displayed using VTK within a Qt widget called the 3D view. This can be seen in figure 3.6.

VTK is a visualisation toolkit [37]. It is an extensive tool with a lot of functionality for visualizing volumetric data. However, in MARS Vision it was only being used to push the final frame onto the screen. The major issue was the path taken by the finished frame. The frame that was generated on the GPU was copied

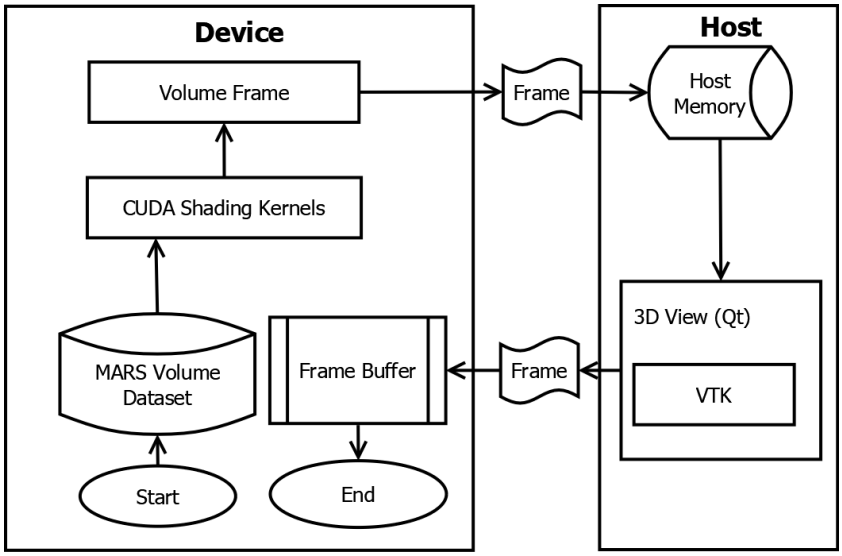


Figure 3.6: A figure showing the original rendering pipeline's flow of actions from the generation of a frame to the frame buffer before it is displayed on the screen.

to the host and then sent to VTK which made further copies before it was sent to Qt. Finally, the frame was sent back to the GPU's frame buffer for display on the screen.

This round-trip taken by the frame costs up to 3 ms of the frame time. Another limitation was that VTK used a time-based trigger to re-draw the screen buffer. This was set to 40 ms. This limited the frame rate to 25 FPS. In order to eliminate this delay, the trigger was modified to push each frame as soon as it was ready. However, the rendering pipeline needed to change to prevent the loss of 3 ms from the unnecessary round-trip for the frame.

To solve this issue, an OpenGL context was introduced into Qt. I made the 3D view use OpenGL for rendering. I initialized a quad with a texture that is stored on the GPU. For volume rendering, I simply used the OpenGL texture mapped to CUDA. The final frame would fill into the texture directly and I simply issued a call to redraw the frame each time it changed. This completely eliminated the 3 ms delay caused by the previous pipeline. Moreover, it also eliminated the need for VTK in MARS Vision as it existed then. The modified rendering pipeline can be seen in figure 3.7.

The default number of iterations for generating one image using the MARS MCRT volume rendering algorithm is 200 iterations. Thus, after my change to

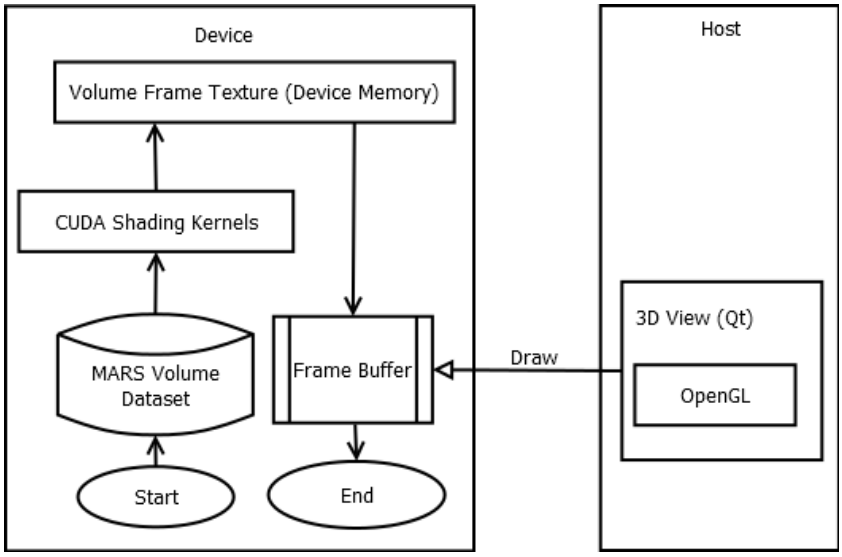


Figure 3.7: A figure showing the modified rendering pipeline's flow of actions from generation of a frame to the frame buffer before it is displayed on the screen. Compared to figure 3.6, note how two copies between the host and the device and no longer needed.

the rendering pipeline that saved 3ms per iteration, which resulted in a total saving of 600 ms. The performance benefit for the MARS MCRT algorithm is mostly negligible since the images already take a relatively long time to render (usually at least one minute). However, it is possible to gain a significant benefit in performance for more real-time rendering algorithms.

3.3 Exploring stereoscopic 3D

The MARS MCRT volume rendering algorithm generates 3D rendered images from MARS datasets. These images can often contain translucent parts based on the transfer function being used. When 3D volume images contain multiple overlapping translucent parts, it can be difficult to explore the image with lighting cues alone.

Stereoscopic 3D provides multiple benefits. One example is binocular depth cues, which improve depth perception while viewing translucent objects. Although there is need for more empirical studies to decisively show the advantage of stereoscopic 3D for CT data, a review by Beurden et al. showed that stereoscopic 3d improved the diagnosis for breast imaging and 3D ultrasound [52].

A review of human performance with stereoscopic displays by McIntire [53] stated that a key advantage of stereoscopic 3D is that it improves spatial understanding of features in 3D objects. This could be very beneficial for the exploration of volumetric datasets. McIntire [54] tried to predict the performance of stereoscopic 3D from the user's level of stereo vision for a 3D positioning task. An interesting finding from this study was that there was no discomfort reported by the users and they found stereoscopic 3D displays comfortable to use. This motivated the need to implement stereoscopic 3D rendering for MARS Vision. This would also enable the implementation of 3D interfaces that rely on stereoscopic 3D displays.

To extend the existing MARS MCRT volume rendering algorithm to support stereoscopic image pairs, the stereo camera positions from the existing camera position were computed. This was done by extending the perspective projection camera to a parallel axis asymmetric frustum perspective projection. The maths described below is based on the geometric transforms described by Southard [55]. This is similar to the approach used in the interactive 3D visualization by Maupu et al. [56]. This can be seen in figure 3.8.

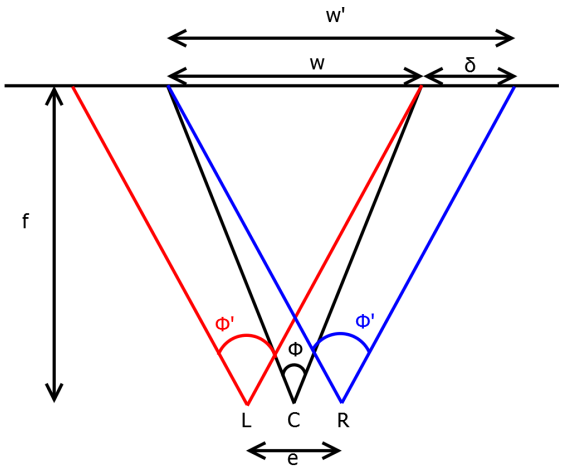


Figure 3.8: Parallel axis asymmetric frustum perspective projection. Where C is the centre camera. L and R represent the left and right cameras respectively. The eye separation in OpenGL units is represented by 'e' and δ is the additional width, which is the same as eye separation in pixels. The focal distance in OpenGL units is represented by f, w represents the width of the screen for the centre camera and w' is the width of the screen for the left and right cameras. ϕ represents the centre camera's field of view.

The first step is to compute the left and right eye camera positions. This can be simply done by defining the eye separation distance e, and moving the camera perpendicularly left by half that distance and right by half that distance. It is more complicated to compute the left and right camera's perspective projection parameters. The parameters that change for the left and right camera are the field of view and the screen width. In order to compute them, a way to convert OpenGL distance to pixels on the screen is necessary, as shown in figure 3.9.

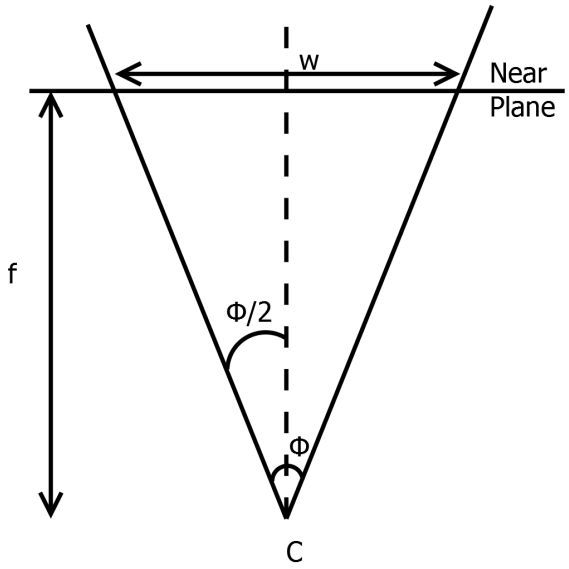


Figure 3.9: Perspective projection of a camera C. f represents the focal distance in OpenGL units, w represents the width of the screen and ϕ represents the field of view.

The figure 3.9 shows the central camera with the near plane, the focal distance f (in OpenGL units), the field of view ϕ, and the screen width w in pixels. Hence, the focal distance f in pixels can be computed by the equation 3.1. Thus, equation 3.2 is derived for converting 1 OpenGL distance unit to screen pixels.

$$f = \frac{w}{2 \times tan(\frac{\phi}{2})} \tag{3.1}$$

$$1\,OpenGL\,Unit = \frac{w}{2 \times f \times tan(\frac{\phi}{2})} \tag{3.2}$$

Equation 3.2 can now be used to compute the delta width δ from figure 3.8. It can be seen from the figure that δ is the eye separation e in pixels. Hence by

substituting equation 3.2, equation 3.3 is derived.

$$\delta = \frac{e \times w}{2 \times f \times tan(\frac{\phi}{2})} \tag{3.3}$$

The new field of view ϕ' is now computed for left and right cameras, by substituting w' and ϕ' in equation 3.1. Thus, the equation 3.4 is derived for computing ϕ'.

$$\phi' = 2 \times tan^{-1}(\frac{w + \delta}{w} \times tan(\frac{\phi}{2})) \tag{3.4}$$

Finally, the left and right camera positions along with the delta width δ, the new screen width w' and the field of view ϕ' can be used for generating the two stereoscopic 3D images using the MARS volume rendering algorithm.

Mesh rendering in stereoscopic 3D

The parameters computed for generating 3D stereo pairs for the MARS volume rendering algorithm can also be used to render meshes. Meshes in MARS Vision are generated quickly using the GPU when needed and rendered using OpenGL. Mesh extraction is discussed in detail in section 3.4. The stereo 3D parameters are converted into view and projection matrices to define the camera for rendering the meshes. These matrices are used for rendering the mesh using the rendering engine discussed in section 3.5.

The left and right camera positions and targets can be used to define the view matrices for the respective stereo images. The projection matrix can be defined using the field of view ϕ' computed in the equation 3.4 along with the new aspect ratio using the new screen width w'. The near and far planes from the centre camera are kept constant and re-used for computing the projection matrix for the left and right eyes.

3.4 Rapid mesh extraction on GPU

With advances in modern graphics hardware, it is possible to rapidly extract iso-surfaces from volumetric datasets. The Marching Cubes algorithm [57] is one of the most popular iso-surface extraction algorithms, however the original approach contained topological ambiguity, which could lead to holes as discussed by

Burkhard [58]. Eventually, the approach from Hansen and Hinker [59] was chosen for the mesh extraction. The approach described was adapted to use General-purpose computing on graphics processing units (GPGPU) with CUDA.

The algorithm proceeds by dividing the volume dataset into a set of cubes. Each cube has eight neighbouring voxels. The algorithm was implemented using CUDA kernels, where each kernel processed a single cube. The user provides the iso-surface threshold as well as the desired voxel size for extracting the mesh.

The voxel size defines the size of the cube and affects the sampling at the corner voxels. For example, when the voxel size is (2,2,2) the corner voxels are factors of 2. In cases where the corner voxel falls between two neighbouring voxels (such as when the voxel size is (1.5,1.5,1.5), tri-linear interpolation is used to compute the value of the corner voxel.

The kernel would compare this threshold with the sampled values at the corner voxels of the cube and classify them as inside or outside the iso-surface. It would then use look-up tables to extract triangles that define the iso-surface within that cube.

The position of the triangle vertices is computed by the linear interpolation of connecting voxels based on the threshold value. The kernel would ignore cubes that lie either completely inside or outside the iso-surface. The flow of actions from threshold definition to mesh generation can be seen in figure 3.10.

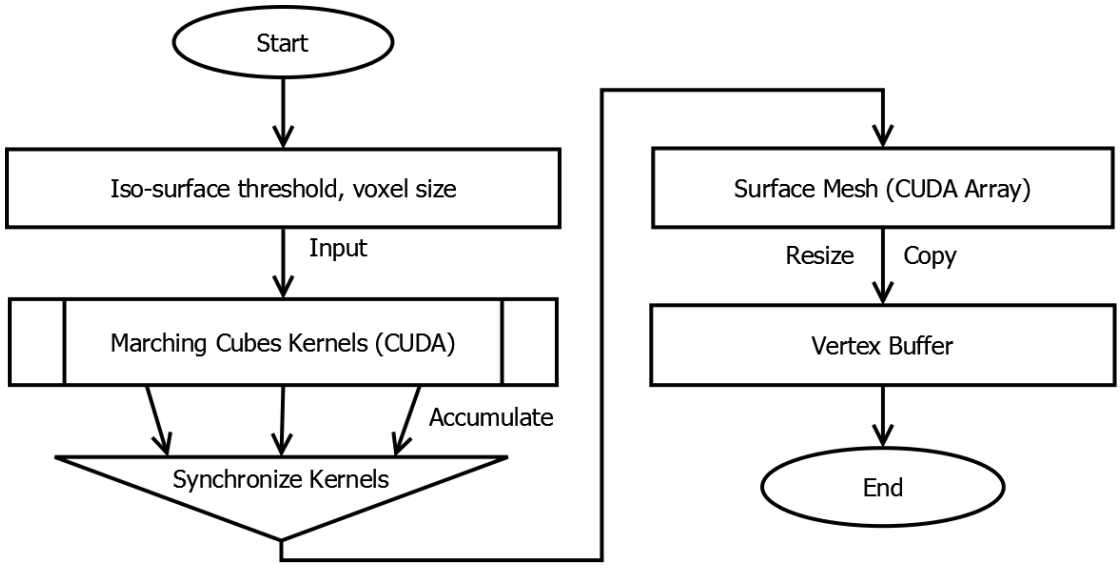


Figure 3.10: Flow actions in the rapid GPU based marching cubes algorithm.

One major issue was that the size occupied by the mesh was only known after the algorithm was executed. Thus, if the mesh size would be larger than the capacity of the GPU's memory, this would result in a crash. Another issue was that once the vertices were generated they needed to be sent to an OpenGL vertex buffer for rendering.

To solve these two issues, the algorithm was modified to work in two passes. The first pass estimates the size of the mesh that will be generated. Later, the necessary space in an OpenGL vertex buffer is initialized and mapped to CUDA. The second pass then generates the triangles and writes the vertices directly into the vertex buffer (see figure 3.11).

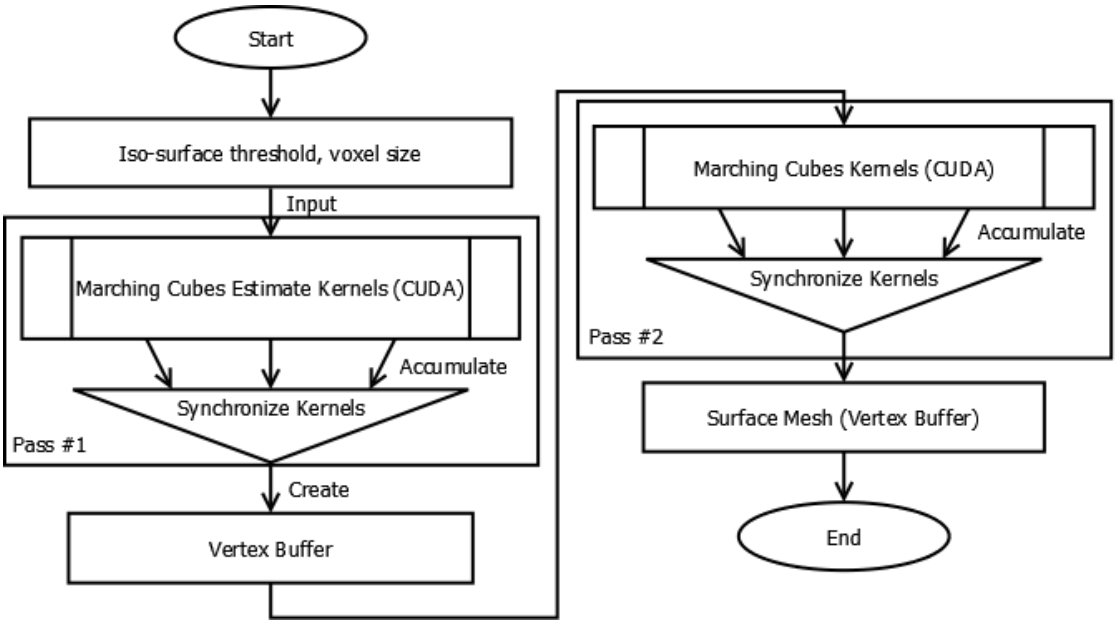


Figure 3.11: Flow actions in the modified 2-pass rapid GPU based marching cubes algorithm.

The effect of the voxel size on mesh quality can be seen in figure 3.12. The iso-surface threshold and the camera orientation are kept constant. The mesh is rendered as a wire-frame where the unit normal vectors are directly mapped to the RGB colour. The voxel size decreases from A to D by a size of (0.5, 0.5, 0.5). Voxel size V and triangle count T are noted in each image's top left and bottom left corners.

This relationship can be represented by equation 3.5. If the mesh is too large

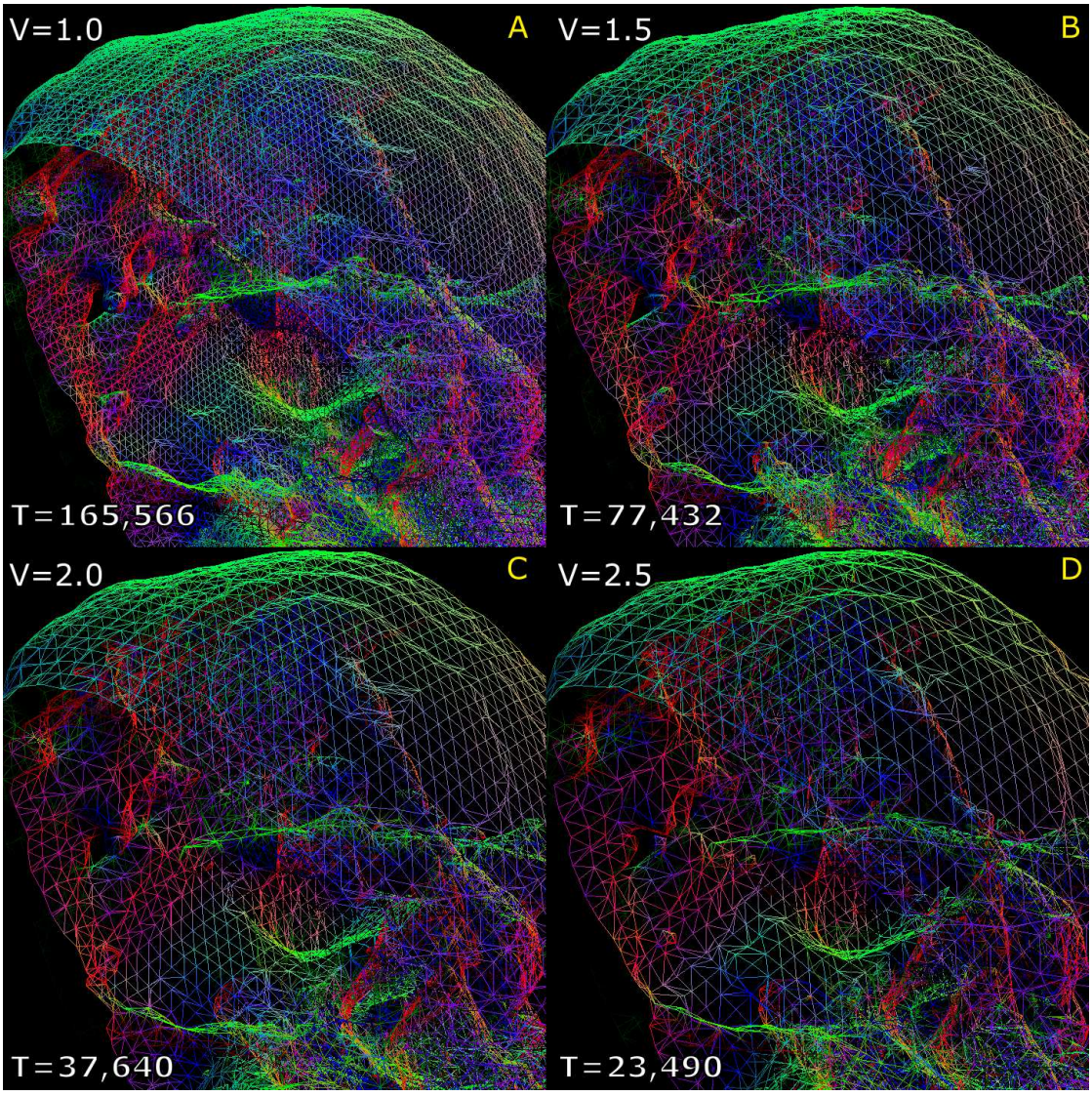


Figure 3.12: A mesh extracted from the MARS crab dataset with different voxel sizes.

to fit on the graphics card, its quality is reduced by increasing the voxel size by (0.125, 0.125, 0.125). This step size equates to an eighth of a marching cube. Based on the typical mesh size for MARS datasets, lower values for the step size do not result in a significant difference in mesh quality but increase the number of steps. Hence, this arbitrary size was chosen.

$$Mesh\,Quality \propto \frac{1}{Voxel\,Size} \tag{3.5}$$

Then, the first pass is executed again. This is repeated until the mesh is small enough to fit on the graphics card. Once the size is appropriate the second pass is executed. This two-pass approach prevents memory crashes while still providing the user with the highest quality mesh that can physically fit in the memory. If the mesh size is small, the user has the option to view more detail (via interpolation) by reducing the voxel size below 1.0.

3.4.1 Extracting multiple meshes from MARS material data

MARS material datasets contain information for multiple material channels, which demand the extraction of multiple meshes. To achieve this, the mesh extraction algorithm is used multiple times to extract a mesh for each material channel. The user is able to modify the iso-surface threshold for one material channel at a time. This affects the corresponding mesh for that material channel. All meshes are stored on the GPU and rendered as multiple 3D models. The meshes can be rendered together as in figure 3.13.

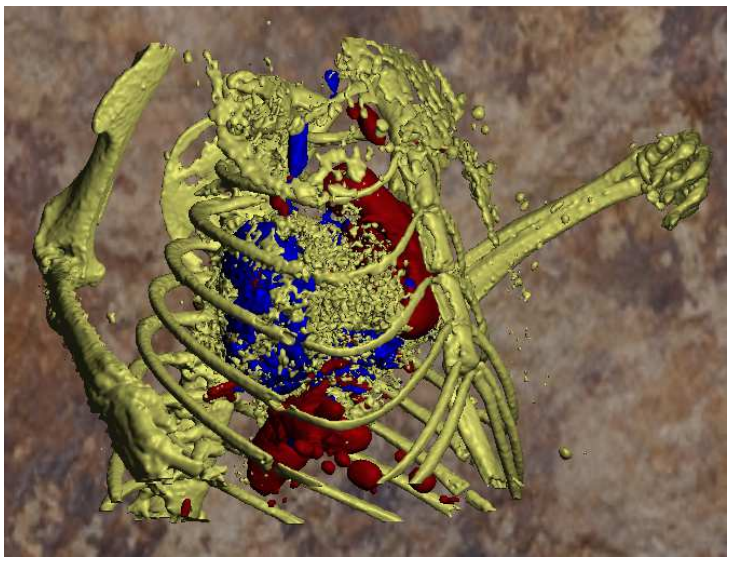


Figure 3.13: Multiple meshes from the Mouse12 dataset channels rendered together, the off-white colour represents the calcium channel, the blue represents the barium channel and the red represents the iodine channel.

Alternatively, the meshes can be split apart and rendered side by side. These channels can be explored separately by the user. This can be seen in figure 3.14,

where the calcium channel is to the left, the iodine channel (red) in the centre and the barium channel (blue) to the right.

Figure 3.14: Multiple meshes from the Mouse12 dataset channels rendered side by side.

Fitting multiple meshes shown in the GPU memory can be a challenging task due to the limited GPU memory. To address this challenge, the size of each mesh is first estimated prior to extraction in phase 1 of the algorithm. If all meshes fit within the available GPU memory, they are extracted, otherwise, the quality of all meshes is reduced until they all fit in the memory. This is done to prevent a large disparity in the quality of meshes. The user has the option to increase the quality of the material channel meshes (one mesh at a time) if there is sufficient memory available or at the cost of the quality of other meshes.

3.4.2 Efficient computation of gradient normals

The Marching Cubes algorithm computes a set of triangles that define the surface mesh. However, the vertex information alone is not sufficient for visualizing the mesh. In order to properly compute the lighting for each triangle, the surface normals are required at each vertex.

For each triangle, the algorithm computes a face normal for shading. This is also written along with the vertex information into the vertex buffer. The surface normal is computed using a cross product of two adjacent edges of the triangle $\vec{N} = Cross(\vec{e}_1, \vec{e}_2)$ (see figure 3.15).

Although this is computationally very efficient, the shading looks jagged. This is due to the fact that irrespective of the geometry, all vertices in a triangle have

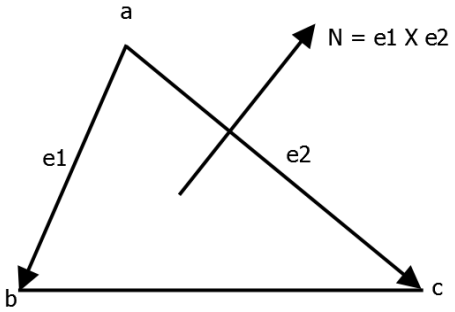


Figure 3.15: Face normal for a triangle.

the same face normal. In order to avoid this, once the face normals have been computed, they need to be smoothed using a mean or median filter [60]. Considering that this needs to be done for about 30 million vertices for a single channel of a typical mesh, it is a very computationally expensive task.

An alternative is to use the information within the volume dataset to compute the normals at the surface. These use a 3D gradient of neighbouring voxels based on the iso-surface threshold, to compute the normal at each voxel. The vertex normals can be obtained by linearly interpolating these voxel normals and then normalizing the result. This is similar to the interpolation done to find voxel positions. The gradient normals are a more accurate representation of the surface normal and the result is much better shading. Once the vertices are computed, the normal at each vertex can be computed using the central difference (gradient) in the X, Y, and Z axes (see equation 3.6).

$$\vec{g} = \frac{1}{2} \begin{Bmatrix} \frac{f(i+1,j,k)-f(i-1,j,k)}{X_f} \\ \frac{f(i,j+1,k)-f(i,j-1,k)}{Y_f} \\ \frac{f(i,j,k+1)-f(i,j,k-1)}{Z_f} \end{Bmatrix} \tag{3.6}$$

The issue with the equation 3.6 is that it requires 6 memory look-ups to compute the gradient normal for each voxel. Memory look-ups are the most expensive operations in GPU processing and these extra look-ups heavily decrease the performance of mesh extraction. The Marching Cubes algorithm already performs eight memory look-ups for the corner voxels in the cube (see figure 3.16).

Given the geometry of the cube, each corner voxel of the cube already contains data from the three neighbouring voxels. Thus, only three additional look-ups are

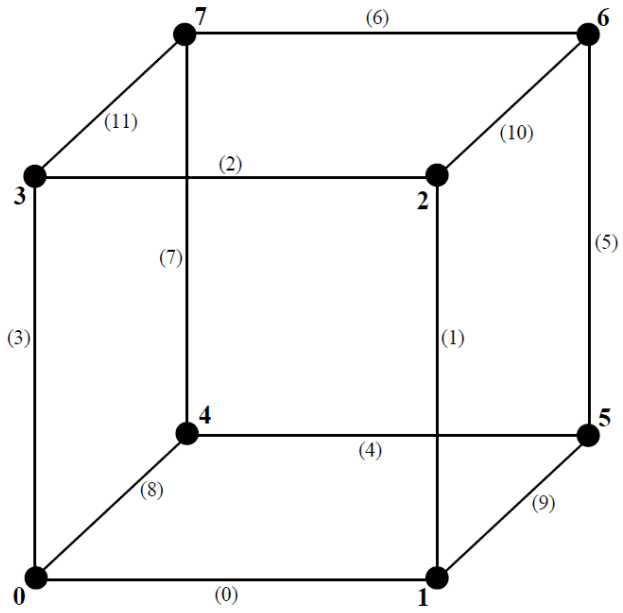


Figure 3.16: A figure showing the marching cube, its corner voxels, edges and the corner-index table.

necessary for computing the gradient normal. In order to re-use the existing voxel data for computing gradient normals, I developed a novel approach alongside the marching cubes algorithm using additional look-up tables.

First, an index for each corner is defined by packing the current corner index into an 8-bit integer. This results in a corner index, as shown in table 3.1.

In order to reuse the corner values, the adjacent x, y and z values are used along with the corner index, see table 3.2. A bit-wise *AND* operation is performed between the corner index and the values. If the result is non-zero, then the corresponding adjacent value is used.

For example, if the corner is *2*, its corner index is *4* and its value is *102* (see table 3.2). If a bit-wise *AND* operation is performed between corner index(*4* [0000 0100]) and the value(*102* [0110 0110]), it results in *4* (0000 0100), which is non-zero. This implies that if the corner is *2*, the adjacent x value represents *x-1*.

Furthermore, the edge index that is computed during the marching cubes algorithm can be reused, to avoid unnecessary gradient vector computations. To

Table 3.1: The corner voxels and their corresponding corner index values

Corner	Corner Index	Corner Index (Binary)
0	1	0000 0001
1	2	0000 0010
2	4	0000 0100
3	8	0000 1000
4	16	0001 0000
5	32	0010 0000
6	64	0100 0000
7	128	1000 0000

Table 3.2: The corner voxels and their corresponding adjacent values.

Voxels	Value		Adjacent Value	
	Integer	Binary	Value	Adjacent
0, 3, 4, 7	153	10011001	X+1	X
1, 2, 5, 6	102	01100110	X-1	
0, 1, 4, 5	51	00110011	Y+1	Y
2, 3, 6, 7	204	11001100	Y-1	
0, 1, 2, 3	15	00001111	Z+1	Z
4. 5. 6. 7	240	11110000	Z-1	

achieve this, the connecting edges for each corner voxel are checked and they are stored in a unique 12-bit integer, as shown in table 3.3. Later a bit-wise *AND* operation is performed with the edge index to determine whether the gradient vector for the current voxel needs to be computed or not.

For example, given the edge index of *1030* for the corner voxel *0*, the connecting value is *265* from the table 3.3. The bit-wise *AND* operation between *265*(0001 0000 1001) and *1030*(0100 0000 0110) results in *0* (0000 0000 0000), which implies that the gradient vector for the corner voxel *0* need not be computed. Similarly the check is done for every corner.

Thus this optimized way of computing the gradient normals reduces the number of lookups required per voxel from 6 to 3. This amounts to a total reduction of look-ups from 48 to 24 per cube. Additionally, unwanted gradient normals are also skipped resulting in additional performance gains. While it incurs a little overhead for the bit-wise *AND* operations, it still results in a significant performance

Table 3.3: The corner voxels and their corresponding connecting edge index values

Corner Voxel	Connecting Edges			Connecting Index	
				Integer	Binary
0	0	3	8	265	000100001001
1	0	1	9	515	001000000011
2	1	2	10	1030	010000000110
3	2	3	11	2060	100000001100
4	4	7	8	400	000110010000
5	4	5	9	560	001000110000
6	5	6	10	1120	010001100000
7	6	7	11	2240	100011000000

improvement for the Marching Cubes algorithm.

3.4.3 Memory optimization for extracted mesh

The mesh extraction described in section 3.4 focuses mainly on efficiency. This results in many redundant vertices for the extracted mesh. I call this the crude or unstructured mesh. The algorithm outputs three vertices per triangle. This results in n-triangle mesh storing $3n$ vertices. Thus triangles with common vertices still store their individual copies of those vertices, which results in multiple redundant vertices being stored for each extracted mesh. These vertices were later rendered using OpenGL's vertex arrays.

The size of a mesh can be estimated from the number of triangles. Each triangle is made up of three vertices and each vertex is made up of a position and a normal (gradient normal, see section 3.4.2). The position, as well as the normal, are stored as 3D vectors, consisting of 3 floating point numbers each. Thus, for a 32-bit floating point system, each vertex occupies 24 bytes of memory. This amounts to 72 bytes of memory per triangle. For example, a mesh with 1 million triangles will occupy 68.67 megabytes of memory.

The typical mesh size for a single material channel of a MARS spectral dataset is approximately 10 million triangles. A dataset with 4 materials will typically consist of 40 million triangles, which results in a mesh size of 2,746.58 megabytes or 2.75 gigabytes. Since MARS datasets can have up to eight material volumes, this would require about 5.5 gigabytes of memory to store the meshes. This is over

Table 3.4: A comparison of unstructured and semi-structured mesh sizes per cube in the best and worst cases.

Case (Triangles)	Unstructured Mesh	Semi-structured Mesh	Difference
Worst case (1)	24 bytes	30 bytes	+6 bytes (+24%)
Best case (5)	120 bytes	78 bytes	-42 bytes (-35%)

and above the memory already required to store the datasets themselves. Hence, it is important to optimize the memory for extracted meshes.

In order to optimize the memory, it is required to eliminate all the redundant vertices. However, this is quite difficult as each cube is processed independently and in parallel, on the GPU. I chose to tackle this issue in two steps. The first step prevented redundant vertices within each marching cube, resulting in a semi-structured mesh. The second step found and removed redundant vertices between cubes from the buffer, resulting in a fully structured mesh.

In order to achieve this, first, the way to store the vertices was changed to allow the referencing of common vertices for triangles. This was achieved by shifting from OpenGL's vertex arrays to vertex buffers [61]. For each cube, only one copy of the corresponding vertices was stored followed by indices that represented the triangles.

Semi-structured mesh

For the worst case scenario, a cube with only one triangle would result in three vertices and three indices for the triangle. This would result in a size of 30 bytes for that triangle. The best case scenario would be five triangles within a cube that would result in using all six edge vertices and 15 indices for the triangles. This would result in a size of 78 bytes for the five triangles. This difference can be seen in table 3.4.

A semi-structured mesh mostly occupies less memory than an unstructured mesh. However, in some cases, the semi-structured mesh can occupy more memory than an unstructured mesh. For example, when a mesh was extracted from the carp dataset [62] with an iso-surface value of 0.80, the unstructured mesh size was 29.164 megabytes and the semi-structured mesh was 30.466 megabytes. In this

case, the latter occupied 4.46% more memory since the iso-surface in most cubes was defined by fewer triangles per cube.

Semi-structured meshes have an additional advantage of improving rendering performance. The GPU uses the triangle indexing information to cache vertices and reuses the cached vertices to improve performance. For example, in the aforementioned carp dataset, the unstructured mesh renders at 88 FPS while the semi-structured mesh renders at 150 fps (+70%). In general, a 30% to 60% gain in rendering performance was observed for semi-structured meshes extracted from MARS datasets.

Fully-structured Mesh

In the second step, the vertex buffer is parsed and redundant vertices are removed. The corresponding indices are reset to point to the first unique copy of the vertex. This results in a significant reduction in the size of the mesh. This step converts the semi-structured mesh to a fully structured mesh with no redundant vertices. This is achieved by searching the entire buffer and storing only unique vertices into a secondary buffer. The triangle indices that reference that same vertex are then modified to point to the unique copy of that vertex. This results in a significant reduction in mesh size.

For example, an unstructured mesh extracted from the carp dataset with the iso-surface threshold set at 0.88, the mesh size it creates is 60.93 megabytes. After converting that to a fully-structured mesh, the size is reduced to 17.41 megabytes. Approximately 70% reduction in memory consumption can be observed, with an additional advantage of improving the rendering performance.

One adverse effect of fully-structured meshes is that they reduce the cache efficiency of the GPU. Since only unique vertices are stored, the indices that reference them can be spread far from each other resulting in more cache misses thereby reducing performance. This can be seen in table 3.5.

Render cache optimization

Optimizing the render cache to improve mesh rendering performance is a well-researched topic. One approach to optimizing indexed triangle strips is a strip growing algorithm aimed to reduce cache misses by Hoppe [63]. Another algorithm

Table 3.5: Comparing sizes and performance of unstructured, semi-structured and fully-structured meshes.

	Unstructured	Semi-Structured	Fully-Structured
	Carp dataset (@ iso 1.07)		
Vertices	1,274,172	851,984	267,769
Size (Mb)	29.16	20.72	7.34
FPS	88	150	120
	MARS Cartilage dataset (calcium channel @ iso 39)		
Vertices	30,647,016	20,349,880	6,578,066
Size (Mb)	701.46	495	179.79
FPS	105	159	135

for improving cache performance for triangle meshes using a first in first out (FIFO) vertex buffer similar to that used in modern GPU's was proposed by Bogomjakov and Gotsman [64]. Later a more efficient algorithm based on greedy optimization for polygonal meshes was proposed by Lin and Yu [65]. They also expand their algorithm to progressive meshes.

A modified version of this algorithm using simulated vertex cache with iterative refinement was proposed by Forsyth [66]. This algorithm was used to improve the rendering performance of the fully structured meshes. A performance improvement of about 26% was observed for fully structured meshes and about 4% for semi-structured meshes. This made the fully structured meshes as efficient as the semi-structured meshes, thereby negating the performance loss.

However, the major issue was the computational cost for the algorithm. The algorithm used for converting semi-structured meshes to fully-structured meshes is not only computationally expensive but also requires three times the memory size of the structured mesh, which limits the scalability of the algorithm. Furthermore, the render cache optimization algorithm added additional computational cost. Due to these issues, the semi-structured meshes without cache optimization was chosen for MARS Vision.

3.5 Real-time rendering engine

MARS datasets can now be visualized using either the MARS volume rendering algorithm or the rapid mesh extraction. It is important to provide a rendering

engine to handle the rendering of both these methods as well as to facilitate rendering other 3D models. This would be necessary for exploration and development of future 3D user interfaces.

Since the MARS volume rendering algorithm was implemented using CUDA, the rendering engine was required to work with CUDA to output the 3D images. Since most commercial rendering engines are designed to handle only 3D polygonal models and have license restrictions, I chose to design my own rendering engine based on OpenGL. The OpenGL CUDA inter-op was chosen to facilitate the integration of MCRT algorithm with the rendering engine. The main motivation was that the MARS project required a rendering engine that did not use external licenses so that the license terms could be made to suit the MARS project.

3.5.1 Overview

The MARS rendering engine was designed with a key focus on performance. The loading of 3D models in the wavefront object (OBJ) format, along with their associated textures was also supported. These models can then be used as 3D user interface elements or for other interaction purposes.

Processing user input is the first step in the rendering engine. The user input may change the viewpoint camera or other volume rendering parameters. After processing the user input, the engine chooses one of two separate rendering paths. One path handles the rendering of 3D volume images while the other handles mesh rendering. These are chosen by the user by selecting the rendering mode.

The volume rendering path begins by updating the camera parameters used in the MARS 3D volume rendering algorithm. This is followed by updating the volume rendering parameters and launching the volume rendering kernels in CUDA. Once the final frame is generated it is rendered on the screen buffer.

For the mesh rendering path, all models including the mesh are stored in a scene render queue. The path begins by updating the view and projection matrices for the scene. Once the matrices have been updated, the physics (collision detection) for all models in the scene render queue, including the mesh, are processed. Finally, all the models in the render queue are rendered on the screen buffer using OpenGL and GLSL shaders.

Both paths result in a final image rendered into the screen's back buffer. Once

the buffer is filled, the front and back buffers are swapped to show the image on the screen. The flow of actions in the rendering engine is shown in figure 3.17

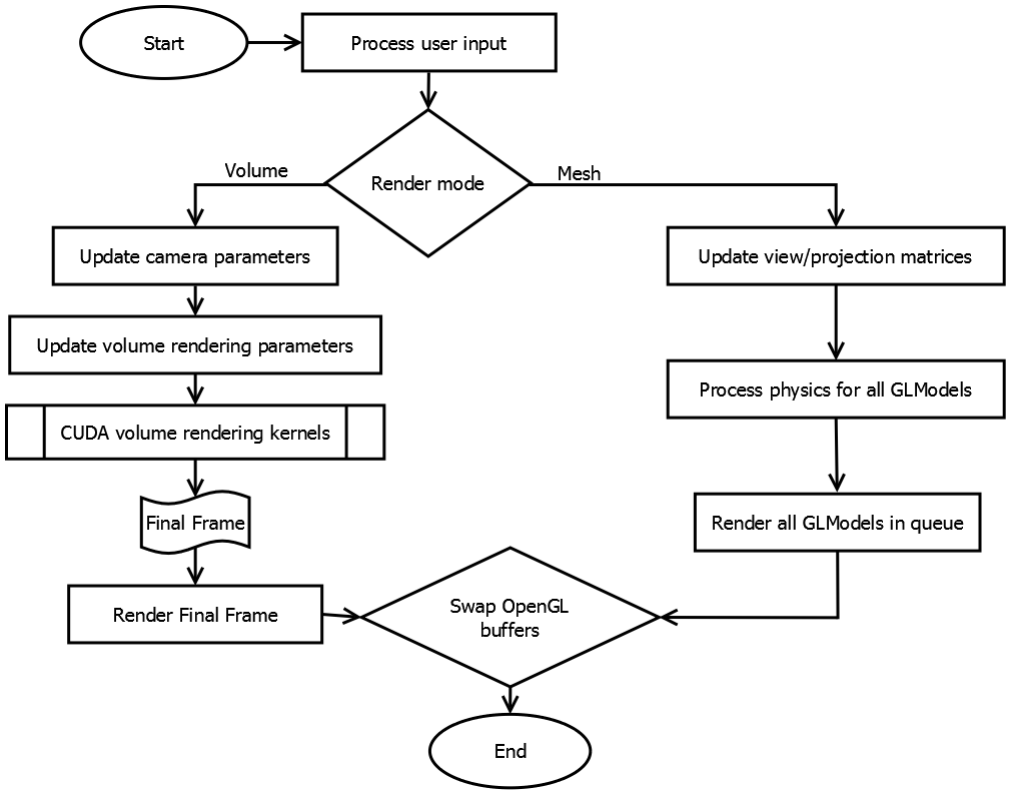


Figure 3.17: The flow of actions in the MARS rendering engine.

The mesh rendering part of the engine is similar to a typical game rendering engine with a render loop [67]. The objects are stored using C++ classes and use 4×4 matrices to store transformations. These are used along with the view and projection matrices to define the model's position. All models are rendered using GLSL shaders. The shaders are based on Phong's illumination model [68]. The lighting model used in the engine for point lights is given by the equation 3.7.

$$I = I_a + \alpha(k_d \sum_{j=1}^{n} (\vec{N}.\vec{L}_j) + k_s \sum_{j=1}^{n} (\vec{N}.\vec{L'}_j)^n) \tag{3.7}$$

Where I represents the illuminated point, I_a represents the ambient light, α represents the light's attenuation, $\vec{N}$ represents the normal of the surface, $\vec{L}_j$ represents the direction vector for the j^{th} light source, $\vec{L'}_j$ represents the vector halfway

between the viewer and j^{th} light source, and n is an exponent that defines the glossiness of the surface. The attenuation α in equation 3.7 can be given by equation 3.8.

$$\alpha = \frac{1}{k_c + k_l d + k_q d^2} \tag{3.8}$$

Where the α is the point light attenuation, k_c is the coefficient of constant attenuation, k_l is the coefficient of linear attenuation, k_q is the coefficient of quadratic attenuation, and d is the distance of the surface from the light source.

The lighting value I from equation 3.7 is used directly for models without any colour. For models with a colour, this value is multiplied with the colour. This includes the extracted meshes. For Models with textures, the final lighting value is multiplied with the texture's colour value.

3.5.2 3D Models

The basic model in the rendering engine is called a 'GLModel'. All 3D models in the scene are extensions of the 'GLModel'. Each model has a unique set of transformation matrices that define its position within the scene. These are used to perform various transformations such as translation, rotation, and scale.

In order to render 3D models in the scene, they need to be loaded from the disk. Once a model is loaded, its corresponding vertices are then compiled into a format suitable for storage in the OpenGL vertex buffer. Later, these vertices are uploaded to the buffer for rendering.

Loading 3D models

The rendering engine supports the loading of 3D models that are stored in wavefront (OBJ) format. Since the format is string based, lines from the file are read one by one ignoring the comment lines. These are then parsed based on the specifiers and stored into arrays representing triangle indices, vertex positions, normals, and texture coordinates.

If the model is textured, the corresponding texture file from material information is also loaded as a bitmap into the memory. For models without textures, the colour information from the model's material is stored. The mesh model is extracted from the dataset and does not require any prior loading.

Compiling 3D models

Once the model is loaded into arrays, it needs to be compiled. The vertices in the array are used for computing the model's centre and bounds. These can later be used for computing bounding volumes for collision detection (used for 3D object selection in section 4.3.3). These bounds can also be used to perform an initial scaling of the model.

Once the model is loaded into arrays, it needs to be compiled. The vertices in the array are used for computing the model's centre and bounds. These can later be used for computing bounding volumes for collision detection. These bounds can also be used to perform an initial scaling of the model.

Later, the vertex information is stored in the GPU using OpenGL vertex buffers. The vertex positions are packed along with the corresponding normals and texture coordinates in a single vertex buffer. The triangle indices are stored in a corresponding index buffer. This pair of buffers is later used for rendering the 3D model.

If the model is textured, the corresponding texture bitmap from the memory is also uploaded to an OpenGL texture. The mesh model does not require any compilation since the extracted mesh is already stored in vertex and index buffers.

Rendering 3D models

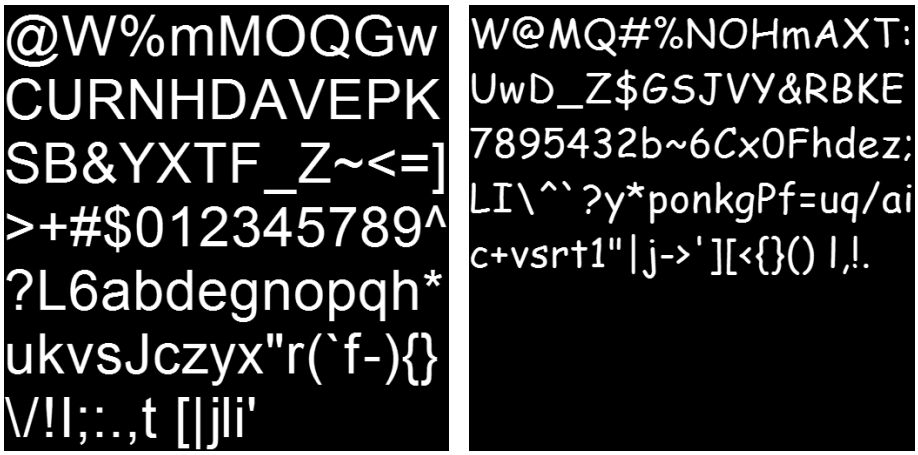
The 3D models including the extracted mesh are stored in a render queue. Later, all models from the queue are rendered on the screen. The queue is also used for processing physics for each model. The models are mostly rendered opaque but can be rendered translucent.

To remove an object from the scene, the corresponding 3D model is simply removed from the render queue. Similarly, models can be added to the scene by adding them to the render queue. The engine makes use of different shaders to handle models with and without textures.

3D text rendering

Rendering text is an important aspect of any engine. Text can not only be used to provide information about 3D models and various UI elements but also to display

annotations and measurements. To render text, the font is first converted to a bitmap texture. Some examples of these font images can be seen in figure 3.18.



(a) Arial (b) Comicsans

Figure 3.18: Bitmap images of fonts used for Text rendering.

The coordinates and sizes of each character are stored in a separate font descriptor file. The font bitmap image is loaded into an OpenGL texture and is used to map the characters to quads in the scene.

Strings of text can be rendered by generating multiple quads in a sequence and textured with the font texture using the respective character's coordinates from the descriptor file. A set of quads need to be generated for each string. The font texture stores white characters on a black background. The white colour can be easily converted to the desired colour. The black background can be used to make the character's background transparent.

The rendering engine provides a unified and efficient method for MARS Vision for rendering the 3D volume image from the MARS MCRT algorithm as well as the extracted iso-surface mesh. Finally, it also allows rendering of 3D models and 3D text, which can be used to provide visual cues to the user and facilitate the design of a 3D GUI.

3.6 Arbitrary slice

Although most clinical diagnosis tasks only involve the use of the three orthogonal slice views, some specialized tasks require the use of an arbitrary slicing plane (also sometimes referred to as the oblique plane). The orthogonal slice views are the standard anatomical planes perpendicular to one of the three axes. The arbitrary slice is not fixed to a certain axis and can have any orientation. This is often used to examine cases where the patient's anatomy is not perpendicular to the orthogonal views due to the curvature of the spine or other physical limitations while performing the scan (such as the patient is unable to lie flat on their back).

At the start of my thesis, MARS Vision did not support an arbitrary slice plane. I added the support for the arbitrary slice plane in both 2D and 3D. However, manipulating the arbitrary slice plane was hard in the 3D case. Other research reinforces this observation that controlling a 2D plane in a 3D scene with a 2D mouse is a difficult task [69]. This is a clear example of the 3D manipulation problem that this thesis aims to solve. This problem is addressed in Chapter 4 and is evaluated for a specialized human diagnosis task in Chapter 5.

This section explores the rendering of the arbitrary slice in the 2D view as well as alongside the volume mesh in a 3D scene. The main focus is the definition of a custom 3D plane to represent the arbitrary slice in a 3D scene. This is followed by the computation of scrolling bounds for this plane within the volume. Later, the arbitrary slice plane is also used to render orthogonal slices in the 3D scene. Finally, the solutions to the orientation and texturing issues for the arbitrary slice plane are explained.

3.6.1 Defining the arbitrary slice plane

In a commercial clinical diagnosis tool such as Inteleviewer [51], the arbitrary slice is created by cloning an orthogonal slice view and then rotating its perpendicular axis (normal). I chose to implement the arbitrary slice in a similar manner. When an orthogonal slice view is cloned, its four corner positions, along with the normal, are used to create an arbitrary slice quad. This slice quad also acts as a model within the scene for the rendering engine.

Once the quad is defined, attributes such as the centre and bounding box are also computed for use with collision detection. The four corner points along with

the texture coordinates for the arbitrary slice texture are compiled into an OpenGL vertex buffer. This can be used for rendering the slice quad in its own 2D slice view or alongside the volume mesh in the 3D scene.

To compute the information within the quad, its corner positions are used to compute the corresponding positions within the volume. These coordinates are then used to look-up the information within the volume and fill the arbitrary slice texture using CUDA.

Once the arbitrary slice quad is defined, it can be rotated to the desired orientation. This results in the corner points being rotated and thus the new positions result in different information being filled into the slice texture. The corner points can sometimes end up outside the volume. In these cases, the portion of the slice outside the volume is rendered black (see figure 3.19).

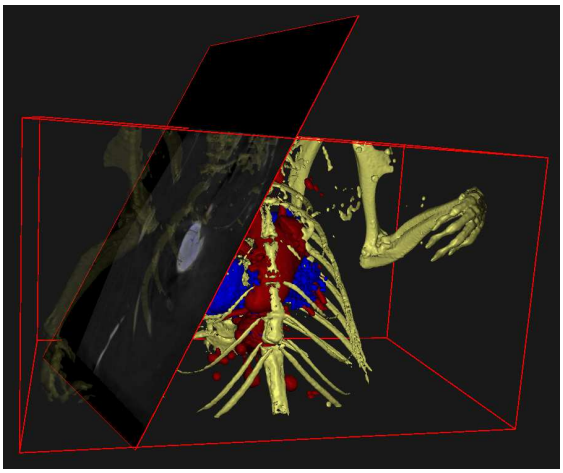


Figure 3.19: Arbitrary slice with the MARS Mouse12 dataset showing the portion outside the volume being textured as black.

3.6.2 Computing bounds and slice thickness

Scrolling the orthogonal slice views is simple since they have a fixed number of slices. However, for scrolling the arbitrary slice, it is important to establish the thickness of the slice along with its bounds. The thickness can be used to divide the bounds into a finite number of slices. The slice number along with the thickness can be used to scroll the slice.

To compute the bounds, a ray with a large magnitude is defined that passes through the centre of the arbitrary slice and faces the same direction as the slice normal. The volume's eight corners are then projected onto this ray. Amongst these projected corners, the min (M_0) and max (M_1) points are chosen as the bounds for scrolling the slice. This ensures that all the information within the volume can be explored by scrolling the slice between the bounds.

This can be seen in figure 3.20, where the white line represents the scrolling bounds for the arbitrary slice.

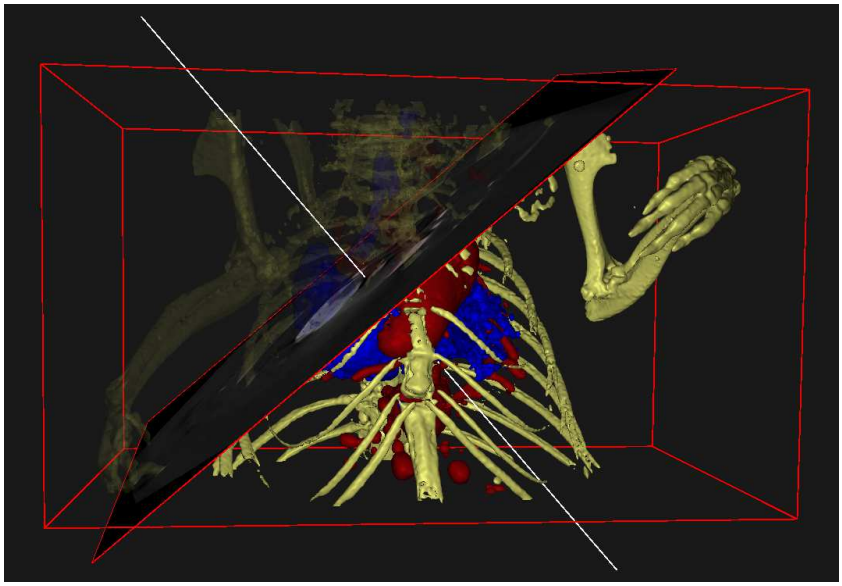


Figure 3.20: Arbitrary slice of the MARS Mouse12 dataset showing the slice bounds (represented by the white line).

The next step is to compute the slice thickness. The slice thickness for a cubic dataset would not change regardless of the direction of the arbitrary slice. However, the datasets are not necessarily cubic. Therefore, to consistently render datasets on the screen they are often scaled by their largest dimension. The scale of the volume is given by equation 3.9, where V_x, V_y, and V_z are dimensions of the volume.

$$V_s = \frac{1}{Max(V_x, V_y, V_z)} \tag{3.9}$$

The slice normal (unit vector) can then be used along with the volume scale

to compute the slice thickness. However, the normal $\vec{N}$ can have negative values depending on the orientation of the arbitrary slice. But the slice thickness cannot be negative. Hence, the absolute value of the normal ($\vec{N_a} = abs(N_x,N_y,N_z)$) is used.

The absolute normal can be projected onto the volume scale to compute the slice thickness Δ_n. This can be done by computing the dot product between the absolute normal $\vec{N_a}$ and the volume scale $\vec{V_s}$ given by the equation 3.10.

$$\Delta_n = Dot(\vec{N_a}, \vec{V_s}) \tag{3.10}$$

Once the slice thickness Δ_n is computed, it can be used to compute the number of slices S_n by the equation 3.11.

$$S_n = \frac{M_1 - M_0}{\Delta_n} \tag{3.11}$$

The current slice number S_c can be computed using the slice's centre position $\vec{C}$ as given by equation 3.12.

$$S_c = \frac{\vec{C} - \vec{M_0}}{\Delta_n} \tag{3.12}$$

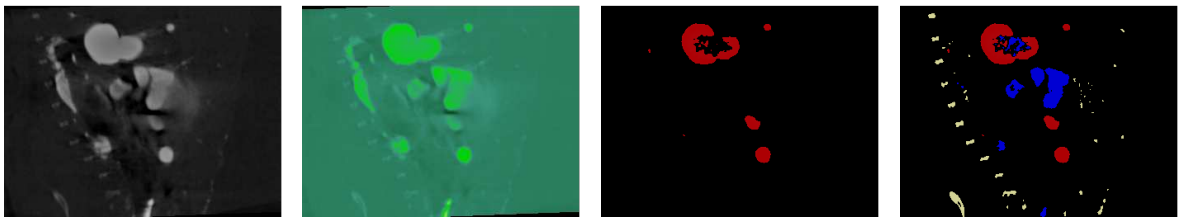
The next slice can be accessed by moving the slice centre by $\Delta_n \times \vec{N}$ and the previous slice can be accessed by moving the slice centre by $\Delta_n \times -\vec{N}$. Similarly, the arbitrary slice centre can be scrolled to the n^{th} slice using the equation 3.13.

$$\vec{C_n} = \vec{M_0} + \left(\frac{n}{S_n} \times (\vec{M_1} - \vec{M_0}) \right) \tag{3.13}$$

3.6.3 Rendering the arbitrary slice

The arbitrary slice quad can either be displayed within a 2D slice view with the same set of controls as the other orthogonal views or rendered as a 3D quad in the scene alongside the volume. The information filled in the texture uses the same algorithm as the one used to render the other slice views. This allows the arbitrary slice view to not only support window and level settings, but also the transfer functions, the look-up tables, and the spectral mode (see figure 3.21).

The arbitrary slice can also be rendered along with the volume mesh in the 3D



(a) Window and level (b) Look-up table (c) Transfer function (d) Spectral mode

Figure 3.21: Render modes of an arbitrary slice from the MARS Mouse12 dataset in the 2D slice view. The attenuation channel was used for a and b, and the material channels were used for c (iodine) and d (iodine, barium and calcium).

scene. This is done by rendering a 3D quad textured with the slice information along with the volume mesh. The portion of the mesh above the slice can be rendered either translucent or completely transparent, to avoid the mesh from occluding the arbitrary slice content (see figure 3.22). The level of transparency can be adjusted by the user. The concept of making the front portion of the volume mesh completely transparent is similar to the plane clipping technique used by Dai et al. [70].

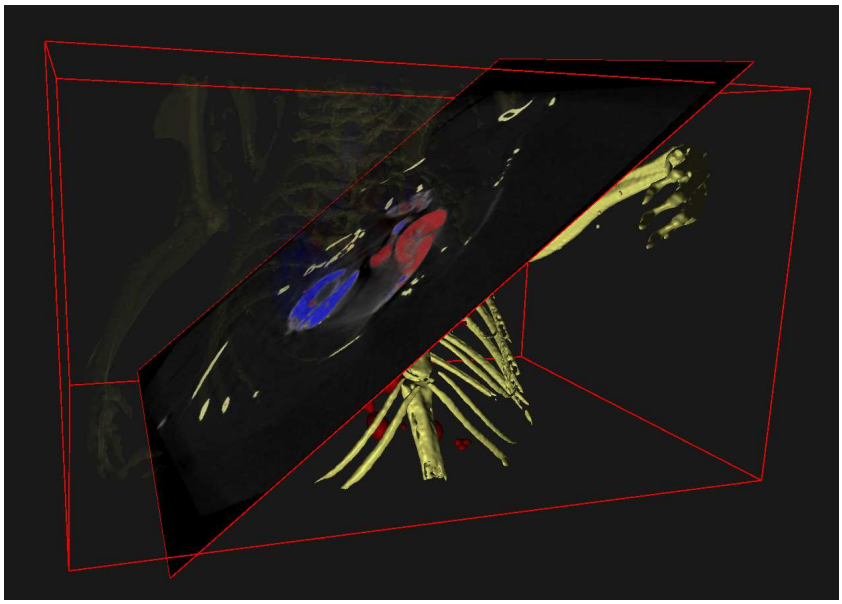


Figure 3.22: Arbitrary slice of the Mouse12 dataset with spectral mode rendered in the 3D scene.

Only the front face of the arbitrary slice quad is textured with the slice information while the other side is simply rendered black. This is done to ensure that the orientation of the slice content in the 3D scene is consistent with the 2D slice view.

3.6.4 Arbitrary slice orientation

The arbitrary slice is created by cloning one of the three orthogonal views. Hence, it is important that when a slice view is cloned, the arbitrary slice view preserves the orientation. The orientation of the information within the arbitrary slice quad depends on the order of its vertices and associated texture coordinates. There are four possible orientations as seen in table 3.6.

Table 3.6: Arbitrary slice orientations

Orientation	Vertex Order
0	0, 1, 2, 3
1	3, 0, 1, 2
2	2, 3, 0, 1
3	1, 2, 3, 0

The arbitrary slice quad is rendered using OpenGL, which defines the front face of polygons by the clockwise orientation of their vertices. Hence, the order of quad vertices is kept constant for all orientation while the starting vertex (bottom left) changes (see figure 3.23).

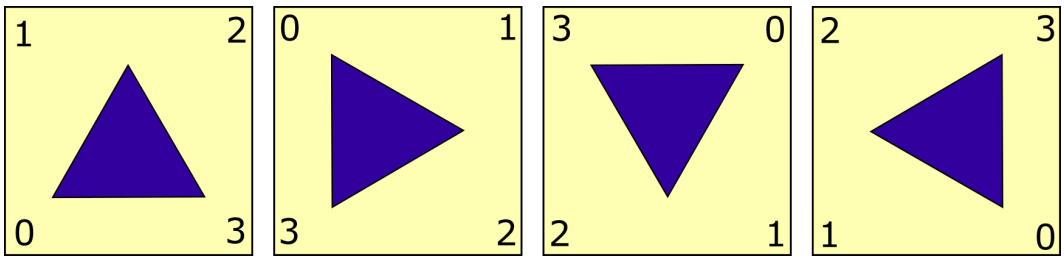


Figure 3.23: Arbitrary slice vertices and orientations from 0(left) to 3(right).

Each orientation shows a different representation of information in the 2D slice view. When an orthogonal slice view is selected for cloning to an arbitrary slice,

the appropriate orientation can be selected. The resulting experience is similar to that supported by commercial DICOM viewers such as Inteleviewer [51].

3.6.5 3D texturing offset

The orthogonal slice information is read directly from the volume dataset and it does not require any interpolation as the slices align perfectly with the voxels. However, the arbitrary slice can be rotated to any orientation and often requires interpolation of voxel data. Thus, the tri-linear interpolation for 3D texture look-up in CUDA is used. However, this can lead to an offset in the information within the arbitrary slice due to the behaviour of 3D texture look-ups in CUDA [71].

The lookup for voxel $V_1(x,y,z)$ results in an interpolation of voxels V_1 and $V_0(x-1,y-1,z-1)$. Thus, when an orthogonal view is cloned, the arbitrary slice displays information that is negatively offset by half a voxel. To solve this issue, a half-voxel offset V_o is added to the texture look-ups for the arbitrary slice. This is given by equation 3.14, where V_s is the voxel scale.

$$\vec{V_0} = \frac{1}{2} \times \vec{V_s} \tag{3.14}$$

3.7 Summary

This chapter discussed the preliminary work for MARS Vision. The key goal was to implement tools to facilitate future research. The improvements to mouse-based manipulation are explained with emphasis on the camera manipulation. The existing algorithm and its limitations are stated along with various improvements to prevent gimbal lock and loss of rotational freedom along with the accommodation for camera panning and clipping planes. Flexible mouse action mapping was also implemented for the 2D slice manipulation.

The MARS MCRT volume rendering algorithm was extended to support stereoscopic 3D rendering. The computation of stereo camera parameters is shown in detail. Later, a method for fast GPU-based mesh extraction in MARS Vision is explained. A novel approach for efficient gradient normal computation for the extracted mesh is also presented. Various memory optimizations for storing the mesh are also discussed that result in approximately 30% size reduction.

Next, the development of a rendering engine for MARS Vision is presented. The rendering pipelines are discussed and an overview of the engine including the selection of the lighting model is given. The process for loading, compiling and rendering of 3D models using OpenGL as well as rendering 3D text in the engine is also explained.

Finally, the implementation of the arbitrary slice is described along with its rendering in the 2D and 3D views. The solutions to orientation and texturing issues for the arbitrary slice are also discussed. This is important as the arbitrary slice is used for the evaluation task in the study described in Chapter 5.

In conclusion, this chapter covers the preliminary work done to enhance MARS Vision in preparation for conducting research into 3D manipulation in medical visualisation software. The work described is all integrated into the current release of MARS Vision provided as part of the MARS product and is used in various sites around the world. In the next chapter, the hybrid interface for better controlling the arbitrary slice is introduced. The rendering engine, mesh and arbitrary slice visualisation, and stereoscopic 3D all form integral parts of hybrid interface aiming to provide an effective workflow for diagnostic use.

Chapter IV

MARS Vision : Hybrid interface development

This chapter aims to provide an effective 3D manipulation technique for the exploration of MARS volumetric datasets. It primarily focuses on the exploration of various input devices for the manipulation of the 3D volume as well as the 3D arbitrary slice. This led to the development of a novel hybrid 2D/3D user interface that aims to improve the efficiency and accuracy for 3D manipulation in exploration and diagnosis tasks. An evaluation of this interface in a human radiology diagnosis scenario is discussed in Chapter 5.

This chapter is structured as follows. Firstly, 3D input devices are discussed in section 4.1. Later, the SpaceMouse interface for manipulating the volume and the arbitrary slice is discussed in section 4.2. The zSpace interface is described in section 4.3. Finally, the 2D/3D hybrid interface development is described in section 4.4.

4.1 3D input devices

3D manipulation is necessary for the exploration of the volume, as well as creating arbitrary slices. This section explores the use of 6 DOF 3D input devices for manipulation. At the start of my Ph.D., the mouse was the only input device used to explore the 3D volumetric images produced by the MARS MCRT algorithm or the extracted volume mesh in a 3D scene. This was done using mouse-based 3D manipulation to manipulate the camera around the volume (see section 3.1.1).

The same method is adapted to map mouse motion to rotate the arbitrary slice about its centre. The user can rotate the arbitrary slice by holding down the [shift] key on the keyboard along with the left mouse button and dragging the mouse in the arbitrary slice view. The difference to the previous method is that the direction of rotation is clockwise as opposed to the anti-clockwise rotation of

the camera.

4.1.1 Comparing the mouse and 3D input devices

Previous research suggests that 3D input devices outperform the mouse for 3D manipulation tasks. A study by Hinckley et al [13] showed that mouse mapped rotation techniques were inferior to integrated 6 DOF devices for 3D manipulation tasks. The study also showed that for 3D rotation tasks, 3D input devices were more efficient than mouse-based methods without sacrificing accuracy. Balakrishnan et al. modified a traditional mouse by adding two additional DOF to form the Rockin'Mouse, which they claim was able to achieve 30% better performance for 3D interaction over the regular mouse [72].

More recently, Scali et al. [73] compared the Phantom haptic interface with the mouse for placing virtual objects on a surface and found that the haptic interface was faster. Schultheis et al. [74] compared a two-handed 3D user interface with the mouse and found that users could interact 4.5-4.7 times faster than the mouse.

Thus, prior research suggests that 3D input devices can provide a better and more natural form of input for virtual cameras (viewport) and object manipulation functions in volume visualisation. They also improve the usability and efficiency of 3D manipulation when compared to mouse-based methods. Hence, I chose to investigate 3D input devices for MARS Vision.

4.1.2 Exploring 3D input devices

Some early 3D input devices, developed particularly for exploring volumetric medical data, include Hinckley's prop system [75], Cutler's responsive workbench [76], and the Cubic mouse [77]. Hinckley's prop system used a tangible prop of a doll's head along with a 6 DOF tracker to enable radiologists to define the desired imaging plane for a CT dataset. Cutler developed an interface using both hands to interact with a 3D model over a responsive workbench using a tabletop virtual reality device. Frolich's cubic mouse consisted of a hand-held 6 DOF cube used to manipulate the three anatomical planes for exploring medical data.

More recently the Wiimote [78] and the Leap Motion controller [79] have been used to capture more natural gesture input. Gallo's interface used the hand-held Nintendo Wiimote controller (6 DOF) as a 3D input device to develop a 3D user

interface for exploring medical data. Mauser's interface used the Leap Motion controller to capture hand gesture input for medical data exploration during surgery.

I had access to several 3D input devices including the Wiimote controller, Kinect (gesture input), Phantom Omni device (haptic pen), Leap Motion Controller, SpaceMouse, zSpace interface, and other tracker based input systems. Since MARS Vision was a desktop application, I chose not to investigate 3D input devices that required a large workspace.

The Wiimote controller, the Kinect gesture input as well as tracker based input devices require relatively large workspaces and are more suitable for immersive environments than desktop environments. While the Phantom Omni 6 DOF haptic device was more suitable for desktops, its mechanical arm made it difficult to be used for continuous 3D rotations. The Leap Motion device used hand gestures to provide 5 DOF for use with desktop applications. While the controller was better than a 2D mouse for 3D pointing tasks [80], a study from Apostolellis et al. [81] found that the 2D mouse outperformed the controller for direct 3D manipulation tasks.

The SpaceMouse input device is similar in size to a regular mouse and can easily be used along with a standard desktop setup. This made it a suitable candidate for MARS Vision. The zSpace interface also consists of a 3D desktop monitor along with stylus input that could easily fit on standard desktops. Hence, I chose to investigate the SpaceMouse input device (see section 4.2) and the zSpace interface (see section 4.3) for 3D manipulation in MARS Vision.

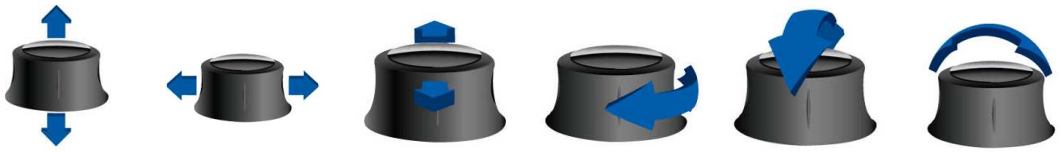
4.2 SpaceMouse Interface

SpaceMouse is an input device developed by 3Dconnexion for 3D manipulation. This device was based on the SpaceBall input device used by NASA to control robots in space. There are many variants of this device, however, the variant investigated in this thesis was the SpaceNavigator [82] (see figure 4.1). The SpaceMouse input device is mostly used in graphics modelling and computer-aided design (CAD) applications.

The device allows the user to perform 3D manipulation tasks such as zooming, panning, and rotation without the need for keyboard shortcuts. The device consists of a pressure sensitive handle over a relatively heavy base. The handle can be

Figure 4.1: A photo of the SpaceMouse device (SpaceNavigator variant).

translated in 3 directions to provide 3 DOF for translation. Additionally, the handle can also be tilted in two directions and rotated like a knob to provide 3 DOF for rotational input (see figure 4.2).



(a) Vertical (b) Horizontal (c) Zoom (d) Spin (e) Tilt (f) Roll

Figure 4.2: 3D manipulations using the SpaceMouse input device. *a* and *b* represent the pan. *c* represents zoom. *d*, *e* and *f* represent rotations.

The SpaceMouse interface uses six different actions to define standard 3D manipulation tasks. The 3 DOF for translation is typically used to control the pan and zoom functions (see figure 4.2). The panning is performed using the up/down action (a) and the left/right action (b). The zooming is performed using the forward/backward action (c). The 3 DOF for rotation is accessed using the spin (d), tilt (e) and roll (f) actions. The spin, tilt, and roll are used to rotate about the Y, X, and Z axes respectively.

The device was initially used to provide intuitive 3D manipulation in virtual environments [83]. A study was conducted by Martins et al. [84] comparing different variants of the SpaceMouse with the standard mouse and keyboard input in a virtual environment. The task involved people with motor disabilities trying to

perform a range of 3D manipulation tasks. The study found that most participants preferred the SpaceTraveller (now SpaceNavigator) variant of the SpaceMouse input device to the standard mouse and keyboard input for 3D manipulation tasks. Hence, I decided to implement the SpaceMouse interface for MARS Vision. The interface setup can be seen in figure 4.3.

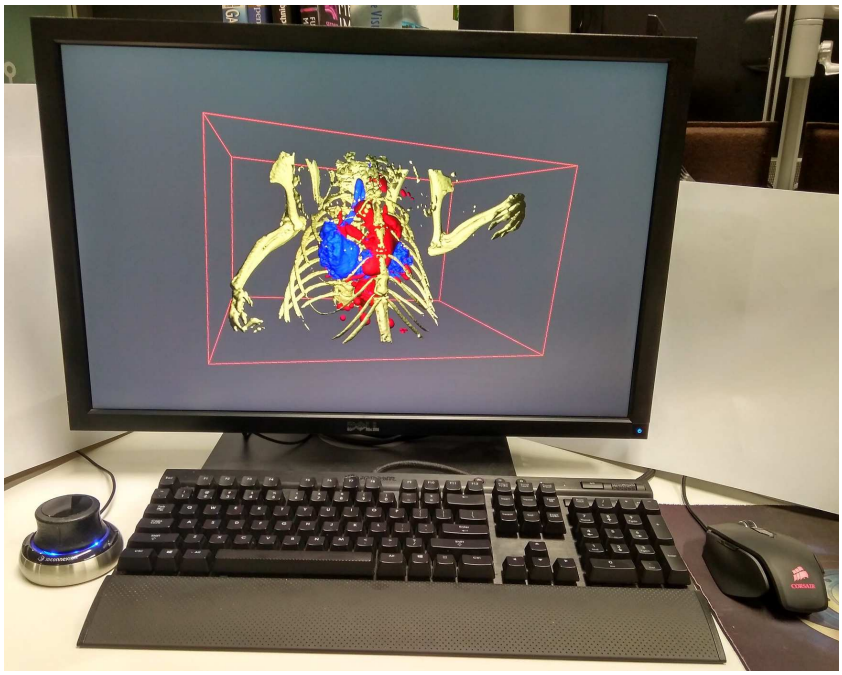
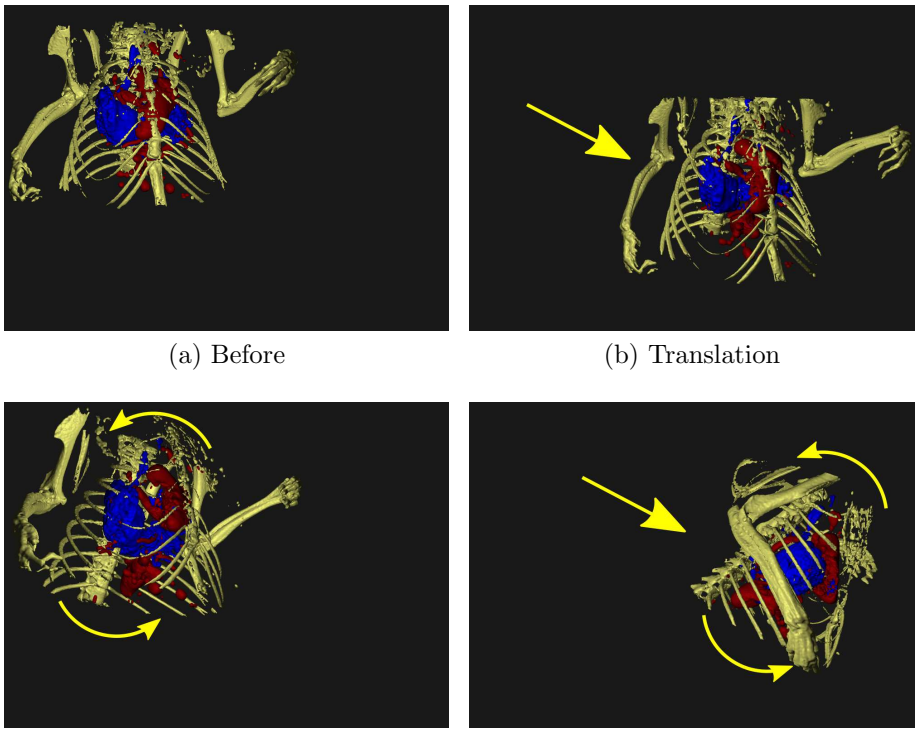


Figure 4.3: SpaceMouse interface for MARS Vision.

4.2.1 Volume Manipulation

The SpaceMouse interface was implemented for volume exploration in MARS Vision. To achieve this, the 3D tasks of zooming, panning, and rotation were mapped to the SpaceMouse actions. When the user rotates the camera to explore the volume in MARS Vision, the camera rotates around the volume in a sphere (see figure 4.4). This behaviour for SpaceMouse interaction is different to that used in most 3D modelling and CAD applications, where the camera functions as a first-person camera that rotates about itself.

The panning and zooming were mapped directly to the camera in MARS Vision. The user could use the SpaceMouse handle translations to pan and zoom



(a) Before (b) Translation

(c) Rotation (d) Translation and rotation

Figure 4.4: SpaceMouse actions to manipulate the camera in MARS Vision.

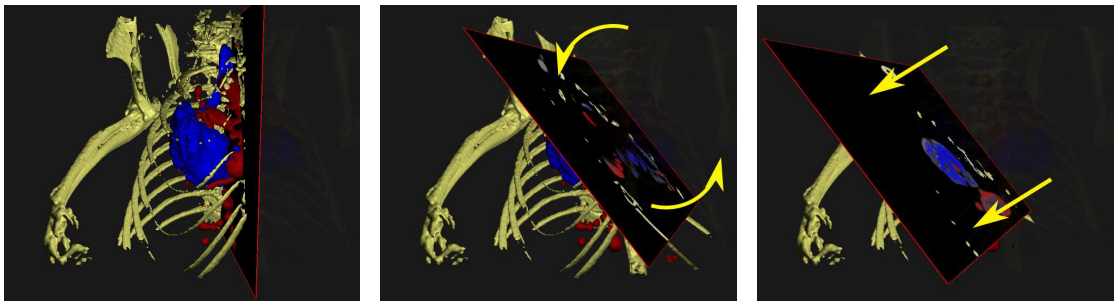
the camera to control the image on the screen. This behaviour is also different to that used in other modelling and CAD tools, where handle translations simply translate the camera in the scene.

4.2.2 Arbitrary Slice Manipulation

Later, the SpaceMouse interface was implemented to manipulate the arbitrary slice in MARS Vision. Unlike the volume manipulation, the 3D translations from the mouse were mapped directly to the 3D translations of the arbitrary slice in the 3D scene. The left/right action moved the slice along the X axis, the up/down action moved the slice along the Y axis, and the forward/backward action moved the slice along the Z axis.

Similarly, the 3D rotations from the mouse were also directly applied to the arbitrary slice (see figure 4.5). The spin rotated the slice about the Y axis, the tilt rotated about the X axis, and the roll rotated about the Z axis. Additionally,

the user could hold the *shift* key on the keyboard and use the forward/backward action to scroll the slice. This would allow the user to move the arbitrary slice perpendicular to its plane.



(a) Before (b) Rotation (c) Scroll

Figure 4.5: Image showing sequential SpaceMouse actions to manipulate the arbitrary slice. *a* is the before state. *b* is the state after rotation. *c* is the state after scrolling.

The SpaceMouse interface was not formally evaluated, however, it was given to the MARS users to gather some feedback. They reported that it was very difficult to use as the handle required very precise movements to manipulate. This resulted in the users either over rotating the camera or accidentally panning or zooming during rotation. It was evident that users required substantial training to precisely use the SpaceMouse for 3D manipulation.

The SpaceMouse was previously implemented for navigation for the real-time MRI exploration by Karin Gardström [85]. He presents an evaluation comparing the SpaceMouse with a 6 DOF tracker based input device called 'Flock of Birds' (FOB), which showed that although participants preferred the SpaceMouse for its haptic feel, they were 14% faster with the FOB. The evaluation also showed that for a more complex task, participants were 70% more accurate with the FOB compared to the SpaceMouse. Based on the feedback from MARS users, as well as the results of the aforementioned study, I decided not to continue with the SpaceMouse and instead find a 3D input device that was intuitive and easier to learn.

4.3 zSpace interface

The zSpace system is a stereoscopic 3D LCD display with embedded cameras that can track the users head and a hand-held stylus for input (see figure 4.6). While most stereo systems use active shutter glasses, the zSpace system uses passive circular-polarized glasses to view the stereoscopic images. This is similar to the passive glasses used for 3D movies in most commercial theatres. This allows the glasses to be very light and comfortable to wear for extended periods of time.

Figure 4.6: An example zSpace system [86] showing the zSpace display along with a user wearing passive stereo glasses and using the stylus. Used with permission.

Given the improvements to spatial understanding from stereoscopic 3D and the intuitive and efficient 3D manipulation from 6 DOF input devices, it became apparent that the zSpace could be ideal for viewing and interacting with 3D content, while being a more intuitive interface for 3D manipulation and annotation than the current desktop system for MARS Vision. This hypothesis is later tested in the Chapter 5.

4.3.1 zSpace head tracking with stereoscopic 3D

Head tracking can be used to create the illusion of a 3D environment and enhance the stereoscopic 3D experience. A stereoscopic display without head tracking simply shows the same image regardless of the user's viewing position. However, when the head position is tracked, the viewpoint can be defined to match the

position of the head. When the user moves his or her head, the viewpoint is adjusted accordingly. This gives the user an impression that the display is a window into a virtual world, instead of a flat image of the virtual world.

This technique is called *Fish Tank Virtual Reality* or *head coupling*. It is possible that the depth perception is better with *head coupling* even when compared to stereoscopic 3D. However, when it is combined with stereoscopic 3D the depth perception gets close to reality, providing a very natural experience [87]. Li et al. [88] compares head coupled perspective and stereoscopic 3D for enhancing applications.

The zSpace interface combines stereoscopic 3D with head tracking. The passive zSpace glasses have five reflective markers (see figure 4.7), which are used to track the position and orientation of the head with respect to the screen. The screen is equipped with infra-red sensors along with infra-red LEDs. The LEDs project infra-red light towards the user which is reflected by the markers. This is used by the sensors to compute the position and orientation of the head.

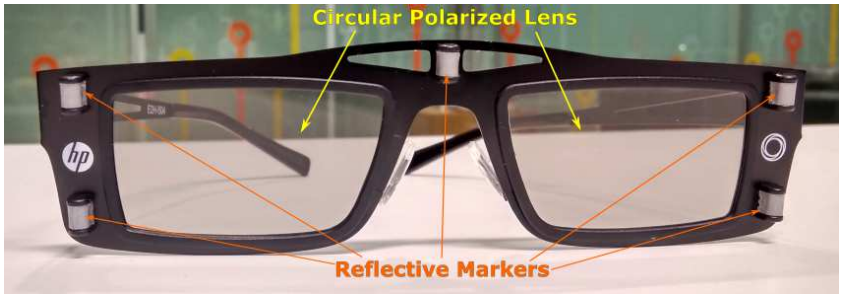


Figure 4.7: Passive circular polarized zSpace glasses showing the reflective markers.

In using the zSpace SDK, the head tracking information is converted to matrices by the zSpace system. For each frame, the zSpace system provides two pairs of matrices to define the user's view and projection for the left and right eye. The scene camera's view and projection matrices are pre-multiplied by the zSpace frustum's view and projection matrices to compute the final view and projection matrices for the left and right eye. These final matrices are used to render the scene to an OpenGL quad-buffer stereo setup (see algorithm 5).

Algorithm 5 Applying zSpace head tracking in MARS Vision's rendering engine. This function is executed for every frame.

Data: Scene camera's view matrix C_v and projection matrix C_p. The zSpace frustum view matrices (left eye F_{lv} and right eye F_{rv}) and frustum projection matrices (left eye F_{lp} and right eye F_{rp}).

Result: Final view matrices (left eye L_v and right eye R_v) and projection matrices (left eye L_p and right eye R_p).

initialization:
Camera view matrix C_v
Camera projection matrix C_p

Render scene
if *render zSpace stereo with head tracking* **then**
  clear buffers
  $L_v \leftarrow F_{lv} \times C_v$
  $L_p \leftarrow F_{lp} \times C_p$
  render back left buffer (L_v, L_p)
  $R_v \leftarrow F_{rv} \times C_v$
  $R_p \leftarrow F_{rp} \times C_p$
  render back right buffer (R_v, R_p)
  swap buffers

end

4.3.2 zSpace stylus interaction

The zSpace system uses a 6 DOF stylus for providing 3D input. The stylus consists of two infra-red LEDs at each end that are tracked by the infra-red sensors on the display. Additionally, the stylus also has three buttons, an RGB LED, and a vibration motor to provide feedback to the user (see figure 4.8).

Mapping stylus motion

To manipulate a virtual object with the zSpace stylus, the stylus motion needs to be mapped to the virtual object in the scene. There are two problems with using the physical stylus to directly manipulate virtual objects. Firstly, although the scene appears 3D, the reach of the physical stylus is limited by the screen. Secondly, the user can easily occlude the physical stylus and the tracking LEDs

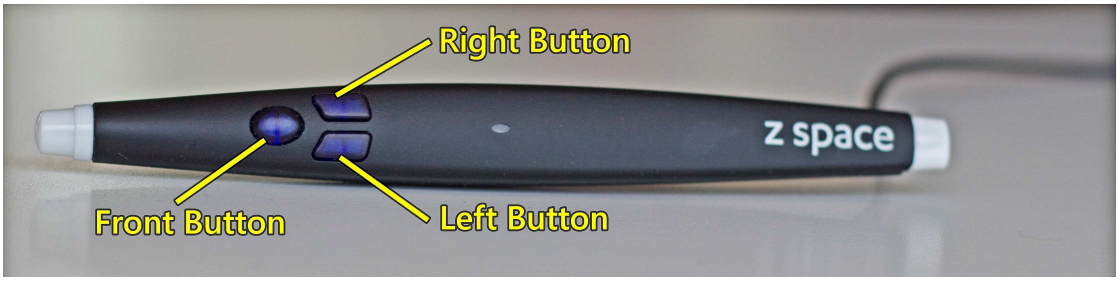


Figure 4.8: zSpace 3D stylus showing its front, left and right buttons. The RGB LED is located at the centre.

during the interaction preventing the tracking from working. Hence, it is important to extend the physical stylus using a virtual stylus to allow users to access objects beyond the physical screen. This requires a GLModel of some sort to represent the virtual stylus in the 3D scene.

The stylus motion from one point to another consists of two components: translation and rotation. These components need to be mapped to quantities that can be used to implement different forms of interaction. If implemented effectively, the stylus motion may then also be logged to provide playback functionality.

The motion of the stylus between two points in time (t_0 and t_1) can be represented using a 3D translation (stored in a 3D vector $\vec{T}$) and a 3D rotation (stored in a quaternion $\mathbb{Q}$). The zSpace system provides a matrix to represent the current pose of the stylus. This matrix defines the position of the stylus in the camera's space. The world pose (transformation) matrix for the stylus is computed by pre-multiplying the inverse camera matrix (see equation 4.1).

$$StylusWorldPoseMatrix = CameraViewMatrix^{-1} \times StylusPoseMatrix \quad (4.1)$$

Once the stylus world pose matrix is computed, it is decomposed into three component vectors: position ($\vec{P}$), direction ($\vec{D}$) and up ($\vec{U}$). Later, the right vector ($\vec{R}$) is computed by performing a cross product between $\vec{D}$ and $\vec{U}$. The difference between stylus position $\vec{P}$ at different points in time is used to map the stylus translation as shown in equation 4.2.

$$\vec{T} = \vec{P}_1 - \vec{P}_0 \quad (4.2)$$

The difference in rotation of each of the three vectors $\vec{D}$, $\vec{U}$ and $\vec{R}$ is stored in what I call a compound quaternion, each consisting of a rotation in two axes. For example, the yaw/pitch quaternion $\mathbb{Q}_{yp}$ can be computed from $\vec{R_0}$ and $\vec{R_1}$ (see equation 4.3). Similarly, the quaternions pitch/roll ($\mathbb{Q}_{pr}$) and yaw/roll ($\mathbb{Q}_{yr}$) can be computed from the D and U vectors respectively. The quaternion function is a well-established function that creates a unit quaternion representing the arc between two vectors [89].

$$\mathbb{Q}_{yp} = Quaternion(\vec{R_0}, \vec{R_1}) \tag{4.3}$$

The rotation about each component axis is stored in what I call the component quaternion. For example, the component quaternion roll can be computed by first rotating the vector $\vec{U_0}$ by the compound quaternion $\mathbb{Q}_{yp}$ and then using this result along with the vector $\vec{U_1}$(see equation 4.4). Similarly, the component quaternions pitch $\mathbb{Q}_p$ and yaw $\mathbb{Q}_y$ can be computed from the component vectors D and R, and the compound quaternions $\mathbb{Q}_{yr}$ and $\mathbb{Q}_{pr}$ respectively.

$$\mathbb{Q}_r = Quaternion(\mathbb{Q}_{yp} \times \vec{U_0}, \vec{U_1}) \tag{4.4}$$

Finally, the quaternion $\mathbb{Q}$ representing the rotation component of the stylus from time points t_0 to t_1 can be derived from the three component quaternions $\mathbb{Q}_r$, $\mathbb{Q}_y$ and $\mathbb{Q}_p$ (see equation 4.5)).

$$\mathbb{Q} = \mathbb{Q}_r \times \mathbb{Q}_y \times \mathbb{Q}_p \tag{4.5}$$

The final rotation quaternion $\mathbb{Q}$ can also be derived from the combination of a component and a compound quaternion. Thus, the equation 4.5 can be simplified to equation 4.6).

$$\mathbb{Q} = \mathbb{Q}_r \times \mathbb{Q}_{yp} \tag{4.6}$$

4.3.3 Stylus interaction styles

Once the physical stylus motion is mapped to translation and rotation components, these can be applied to the virtual stylus to design the interaction with the virtual objects in the scene. Since the user holds the physical stylus and interaction is done

using the virtual stylus, it is important these two appear connected to the user. To achieve this, the head tracking (see section 4.3.1) can be used, which allows for a seamless transition from the physical to the virtual stylus (see figure 4.9).

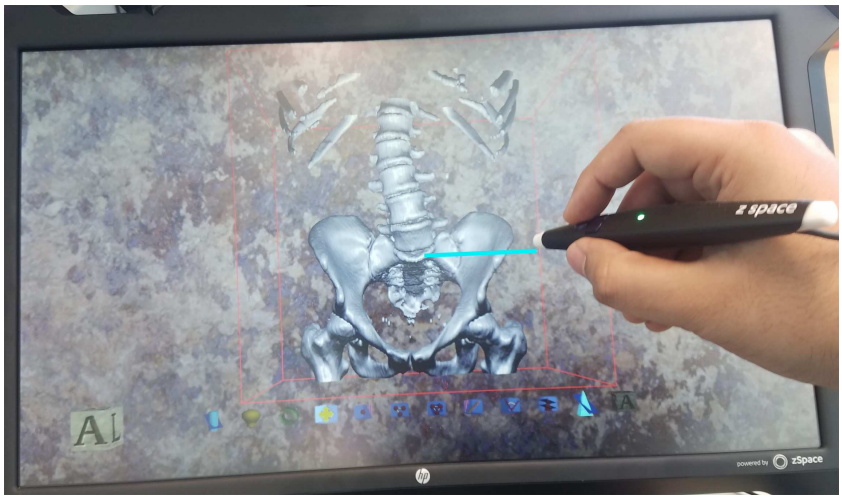


Figure 4.9: An photo from the left eye of passive zSpace stereo glasses with head tracking showing the seamless physical and virtual stylus. Note that the virtual stylus (enhanced for visibility) appears seamless when viewed through the stereo glasses.

The virtual stylus length plays a key role in the interaction. Since the user holds the physical stylus similar to a pen, the stylus precision decreases as the virtual stylus length increases away from the hand. A long virtual stylus length also amplifies any hand tremors, making it difficult to perform precise 3D manipulations.

Based on the behaviour and length of the virtual stylus, two styles of interaction are considered: wand and laser. The styles define how the stylus is used to interact with virtual objects in the 3D scene. In simple terms, the wand style consists of a fixed-length stylus while the laser style consists of a varying length stylus (similar to a laser pointer).

In this implementation for both styles, the bounding volume for the selected object is also rendered in a red wire-frame to provide visual feedback for the user.

laser style

The laser style interaction consists of a virtual laser at the end of the physical stylus. It is similar to a ray-casting interaction [90] with a very large laser length

(seemingly infinite). The user points the laser at an object to select it. When an object is selected with the laser, its length is set to the distance between the physical stylus and the collision point of the selected object (see figure 4.10). This length is fixed while the user manipulates the object with the stylus. Once the manipulation is done, the stylus length is reset.

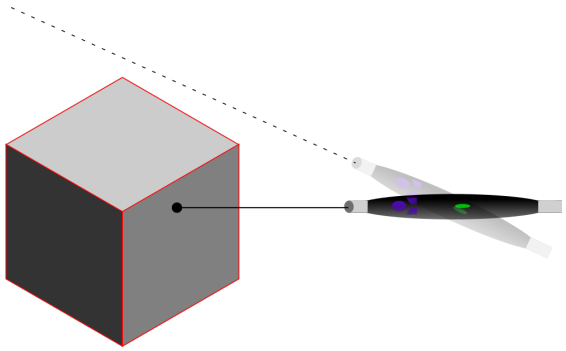


Figure 4.10: laser style stylus interaction.

The laser style interaction uses a ray-to-bounding volume test to detect collision with the virtual objects. The ray is defined by the end of the physical stylus as the origin, pointing in the direction of the stylus away from the hand, with a very large (seemingly infinite) magnitude.

The amount of hand movement required to access all virtual objects is minimal with the laser style interaction. This style allows for easy selection of an object by simply pointing the stylus at them. However, selection of an object that is behind another object is difficult since the laser snaps to the nearest object. This is also a serious problem when objects are located inside other objects such as one mesh inside another (MARS material channel meshes), annotation spheres inside the volume mesh and so on. Due to these limitations, I chose not to use the laser style interaction for the evaluation study.

Wand style

The wand style interaction consists of a virtual wand with a fixed length at the end of the physical stylus (see figure 4.11). Although the wand length remains constant throughout the interaction, the user is able to adjust the length between interactions by accessing the settings dialog. The user can select objects using the

wand and manipulate them as desired.

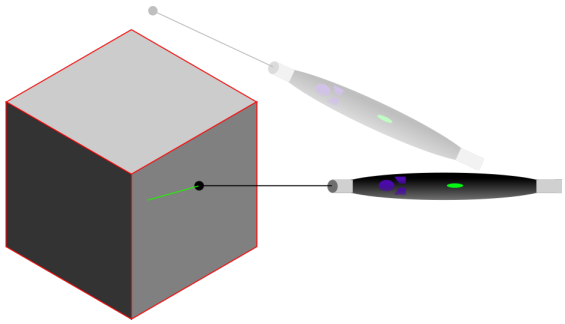


Figure 4.11: Wand style stylus interaction.

The wand style interaction primarily uses the sphere-to-bounding volume test to detect collision with the virtual objects. Since the end of the wand is relatively small, selecting thin objects such as the arbitrary slice can be challenging. To make it easier to select thin objects, the ray-to-bounding volume test is also used to allow a small range of selection area behind the tip of the virtual wand. The ray is defined similarly to the laser style except for the magnitude, which is defined by the wand length.

The wand style requires relatively more hand movement to access all virtual objects in the scene due to the fixed virtual stylus length (which is often relatively small compared to the laser style). This can lead to hand fatigue during prolonged use. However, the fixed length makes it easy to select an object that lies behind or inside another object. Thus the wand style appears to be more useful for the expected scenarios in medical imaging where a collection of objects may occlude one another (volumes, clipping objects, annotations, etc.). This is also why the wand style was chosen for the hybrid interface (see Chapter 4).

4.3.4 Object manipulation with the stylus

Once the user has selected a virtual object using the stylus, the object can be manipulated in various ways. Three types of stylus interaction for manipulating the selected object are implemented in MARS Vision: direct manipulation, 3D translation, and tethered rotation.

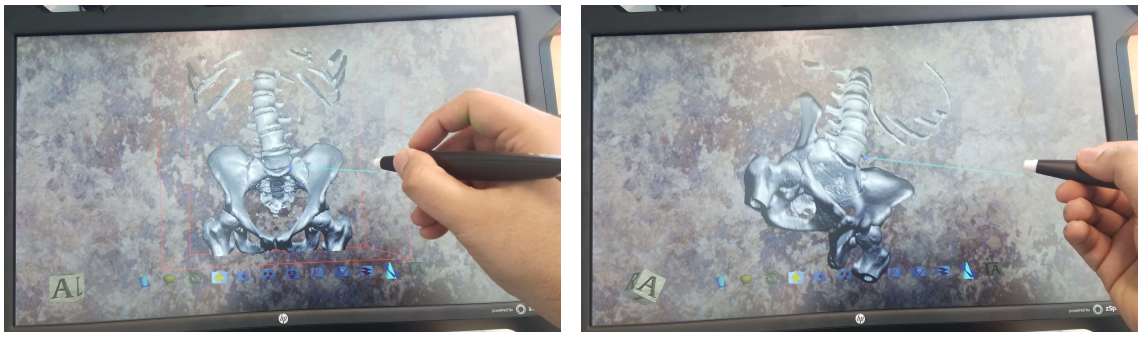
The direct manipulation is a one to one mapping of stylus motion to the selected virtual object (as if it were an extension of the user's hand). The 3D

translation allows the user to translate the selected object while maintaining its orientation (non-rotatable, but movable object). The tethered rotate is implemented to allow the user to perform more restricted and precise rotations on the selected object(immovable, but rotatable object).

Direct manipulation

Direct manipulation in user interfaces was introduced by Shneiderman in 1982. According to him, direct manipulation consists of representing objects of interest with rapid incremental reversible actions [91]. Direct manipulation has since become widely used in most user interfaces.

In the system with the zSpace stylus, direct manipulation allows the user to manipulate the virtual object in a similar way to a physical object. While an object is selected, all motion from the stylus, including translation and rotation, is mapped onto the object (see figure 4.12). During manipulation, the object appears to be attached to the end of the stylus and stays in the same position and orientation relative to the stylus.



(a) before (b) after

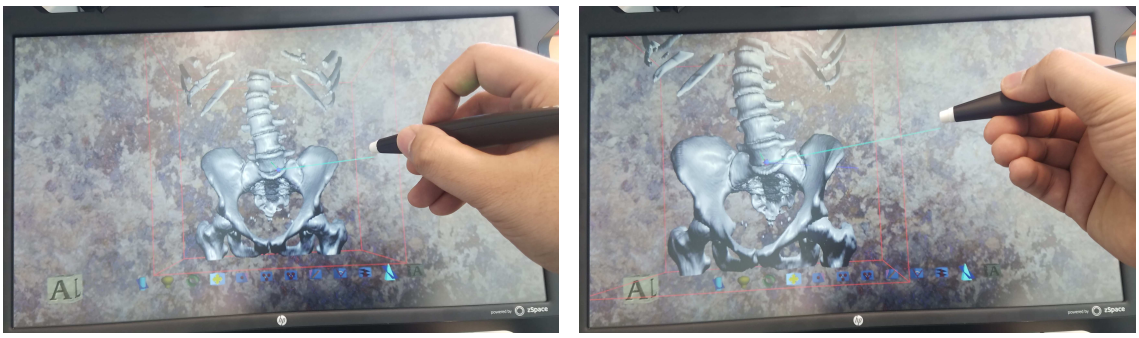
Figure 4.12: Direct manipulation with zSpace stylus.

Direct manipulation is intuitive since the motion in the virtual world is a direct one to one mapping of the user's hand motion in the real world. This also makes the interaction very easy to learn, since users only rely on their hand motor skills. Direct manipulation can have considerable advantages for the manipulation of simple discrete interface elements such as the volume mesh in MARS Vision.

However, research suggests that it is not suitable to perform constrained and precise manipulations [92].

3D Translation

3D translation is a form of direct manipulation where only the translation is mapped onto the virtual object (see figure 4.13). This ensures that the orientation of the object is preserved during the manipulation. 3D translation is especially useful with the arbitrary slice, where users often prefer to preserve the orientation while they move the slice.



(a) before (b) after

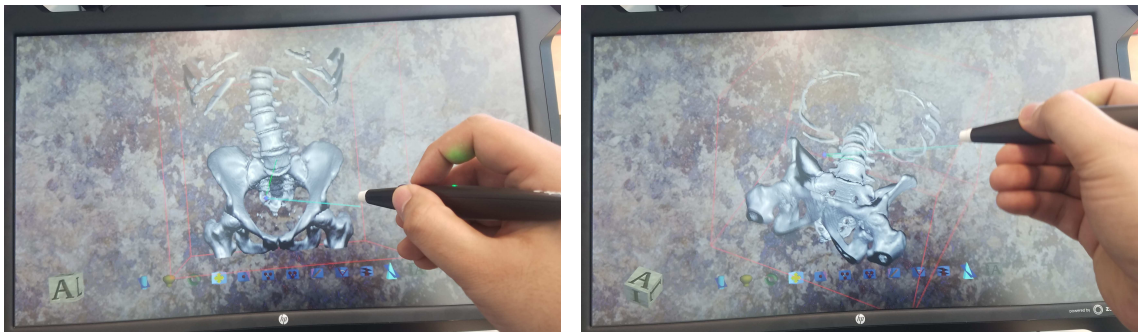
Figure 4.13: 3D translation with the zSpace stylus.

Tethered rotation

Tethered rotation is an interaction technique I developed to allow for more precise rotations. When an object is selected, a tether (3D line) is created from the end of the virtual stylus to the centre of the virtual object. This tether is used to rotate the object. The tether can be rotated in a sphere about the centre of the object. The object's orientation stays relative to the tether (see figure 4.14).

Unlike the direct manipulation approach, for tethered rotation, the object stays fixed in the same place and rotates. Thus, the tethered rotate requires the user to move the stylus in an arc around the object to rotate it. This allows the user to perform more precise rotations.

Recently, I discovered a similar technique used by Saalfeld et al. [93] in their medical viewer. Their approach stores the tethered point in the first step, which



(a) before (b) after

Figure 4.14: Tethered rotation with the zSpace stylus.

is used to rotate the object in a subsequent step. Their interface also used the zSpace stylus for interaction. However, they used a fully immersive head-mounted display, instead of the zSpace stereo display.

4.3.5 3D Tools

This section explores a series of tools using the zSpace stylus for exploring the volumetric data in MARS Vision. These tools are based on the stylus interaction techniques and object manipulation techniques discussed in previous sections (4.3.3 and 4.3.4).

Firstly, the heads-up display (HUD) used to access all the tools along with the common access techniques is discussed. Later, various forms of feedback from the HUD are explored. Finally, the functionality of each of the 3D tools is discussed in detail. These tools can be classified into five categories: object manipulation, slice manipulation, light source manipulation, annotation, and measurement.

The stylus contains three buttons: front, left and right buttons, see Fig. 4.8. The front button is always used for picking and manipulating virtual objects in the scene such as the volume model or the anatomical plane, as well as activating all the 3D buttons. The functionality of the left and right buttons is contextual to the selected 3D tool.

This stylus button interaction was based on the design guidelines laid out in the zSpace manual (see figure 4.15). When the stylus is held as specified by the manual, the user's index finger naturally rests on the front button.

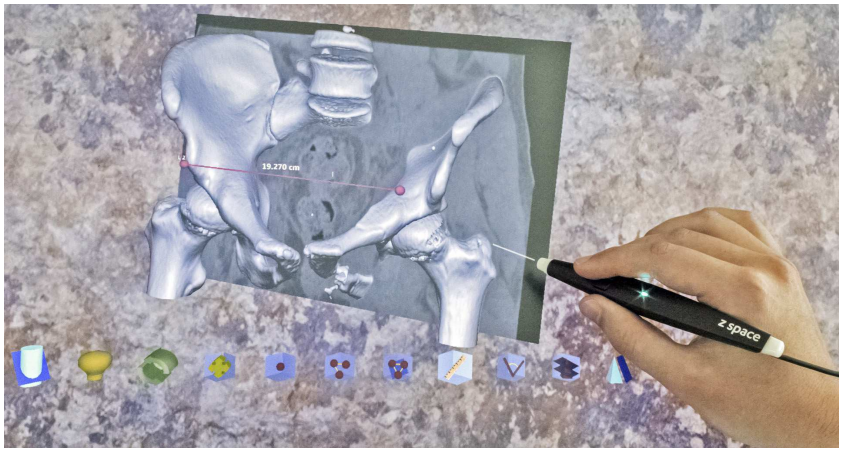


Figure 4.15: zSpace interface for MARS Vision showing the preferred way to hold the stylus.

Hence the most common action (the object manipulation) was mapped to the front stylus button. The left and right buttons had contextual actions based on the interaction mode. For example, insertion of objects (such as annotations or measurements) was mapped to the right button and the deletion was mapped to the left button. A key reason behind this decision was that the right button was easier to reach than the left button for a right-handed person.

zSpace HUD

The zSpace heads-up display (HUD) was designed using familiar principles from 2D interaction, which makes it intuitive and allows users to quickly adapt to it [94]. The HUD consists of a set of buttons (3D widgets) centred along the bottom edge of the screen. Each button is a 3D model that is designed to represent the associated functionality. Although the buttons are 3D models they are placed on the screen at the depth equal to the zero parallax plane, which is similar to 2D buttons in a 2D interface. The buttons in the HUD can be seen in figure 4.16.

When hovering over a button with the stylus it is highlighted by being slowly enlarged (scaled up), brightly lit, and the associated label is displayed underneath (see figure 4.17). When the user leaves the button by moving the stylus away, the button slowly returns to the default scale, its lighting is reset to normal and the label is hidden.

Figure 4.16: Image showing all the buttons in the zSpace HUD. They include the following from left to right of the image: slice toggle (a), light source toggle (b), reset (c), object manipulation mode (d), point annotation mode (e), line annotation mode (f), three point slice placement (g), line measurement mode (h), angle measurement mode (i), slice oscillation (j), slice attachment toggle (k) and the orientation cube toggle (l).

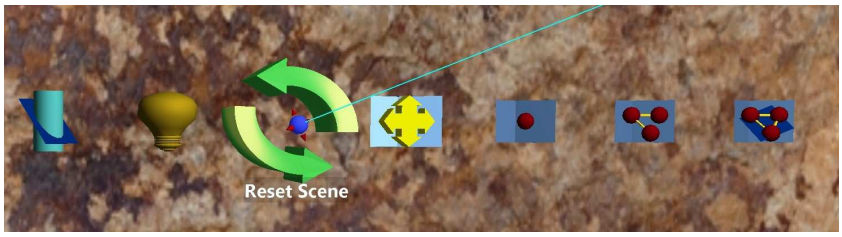


Figure 4.17: zSpace HUD showing how the selected button is enlarged and brightly lit with its label displayed underneath.

Additionally, a translucent billboard can be displayed when a button is highlighted. This billboard shows the stylus buttons and the mapped functionality related to the selected button (see figure 4.18) like a tooltip from a standard 2D GUI. This feature is particularly helpful for new users and can be turned off by the user. It can also auto-hide.

The user can activate the 3D button by clicking the front button on the stylus, while the 3D button is selected. This will trigger the function associated with the 3D button. There are two types of buttons: action buttons and toggle buttons. Action buttons perform a single action when triggered (for example, the reset button) like a push button in regular 2D GUIs. Toggle buttons are used to toggle some states (for example, the toggle slice button shows or hides the slice) like a check button in regular 2D GUIs. Toggle buttons can also be used to toggle between two states (for example, the slice attachment button toggles between attached and detached state). Additionally, the toggle buttons can also be used to represent modes. When one mode button is toggled on, all the others in the group are toggled off. This is similar to radio-buttons in regular 2D GUIs.

The lighting and scale for the toggle button are changed to provide visual feed-

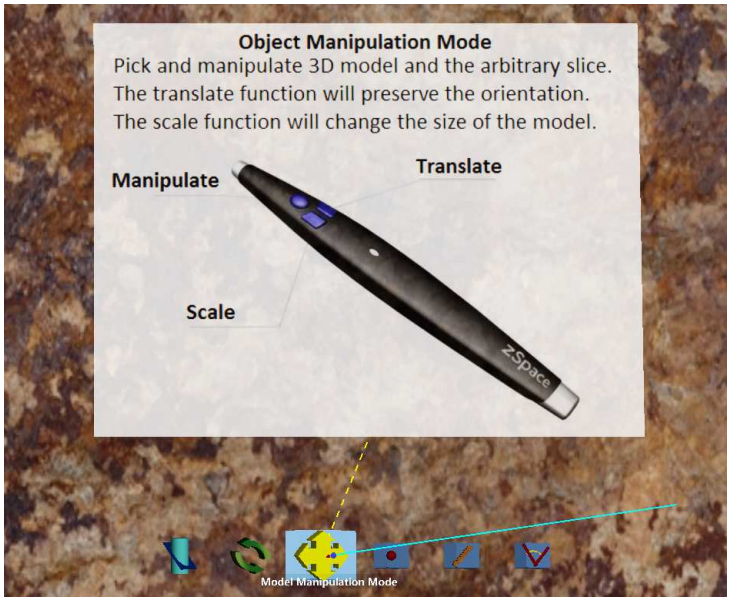
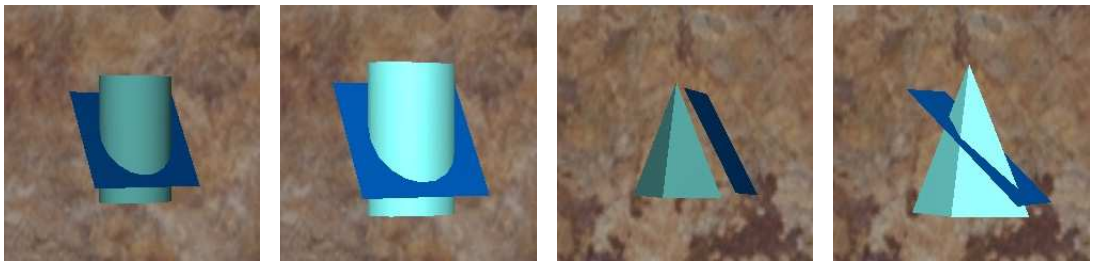


Figure 4.18: zSpace HUD showing how a billboard is displayed when a button is highlighted with the stylus.

back to the user. The button is scaled up and brightly lit to show a *toggle on* state and the lighting is reset back to normal for the *toggle off* state, as shown in figure 4.19. This is also used to highlight the currently selected mode. Alternatively, for some toggles, in addition to the aforementioned visual cues, the 3D model also changes to reflect the current state (see figure 4.19).



(a) Slice off (b) Slice on (c) Attachment off (d) Attachment on

Figure 4.19: Visual feedback from toggle buttons. Notice the scale and lighting change for the toggle slice button from (a) to (b). The model for the toggle slice attachment button also changes in (c) and (d).

When the user interacts with the HUD, visual feedback is provided to the user

in the form of scaling up the buttons and changing the lighting. In addition to the visual feedback, the RGB LED and the vibration motor in the stylus are also used to provide feedback to the user. The stylus LED is used to represent various interaction modes (see table 4.1). The stylus vibrates for 100 ms when the user hits the bounding volume of a virtual object (such as a button or the volume mesh).

Table 4.1: The LED colours and associated modes

LED Colour	Mode
Green	Object manipulation
Red	Point annotation
Yellow	Line annotation
Blue	Three point slice placement
Cyan	Line measurement
White	Angle measurement

In addition to the buttons, the HUD can also be used to display useful information such as the orientation cube. It can be seen in the bottom left corner of the figure 4.20.

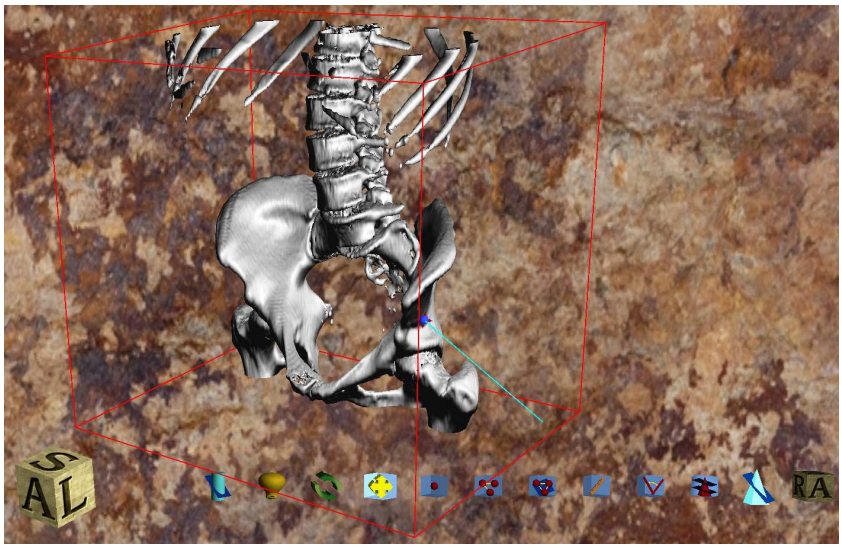


Figure 4.20: zSpace HUD orientation cube in the bottom left. Its orientation matches that of the volume.

The orientation cube is inspired by how 3D volume orientation is represented in commercial DICOM viewers. The faces of the cube consist of a letter representing

the anterior, posterior, top, bottom, left, and right orientations. When the user rotates the volume, the cube also rotates accordingly. This helps the user quickly understand the orientation of the volume being observed.

Object manipulation

Some objects in the 3D scene can be picked and manipulated using the stylus. These include the volume mesh, the arbitrary slice, the light source, annotation points and measurement points. Other objects such as the 3D buttons and the orientation cube cannot be manipulated by the user. The object manipulation mode is the default interaction mode and allows the user to manipulate, translate, or scale the objects.

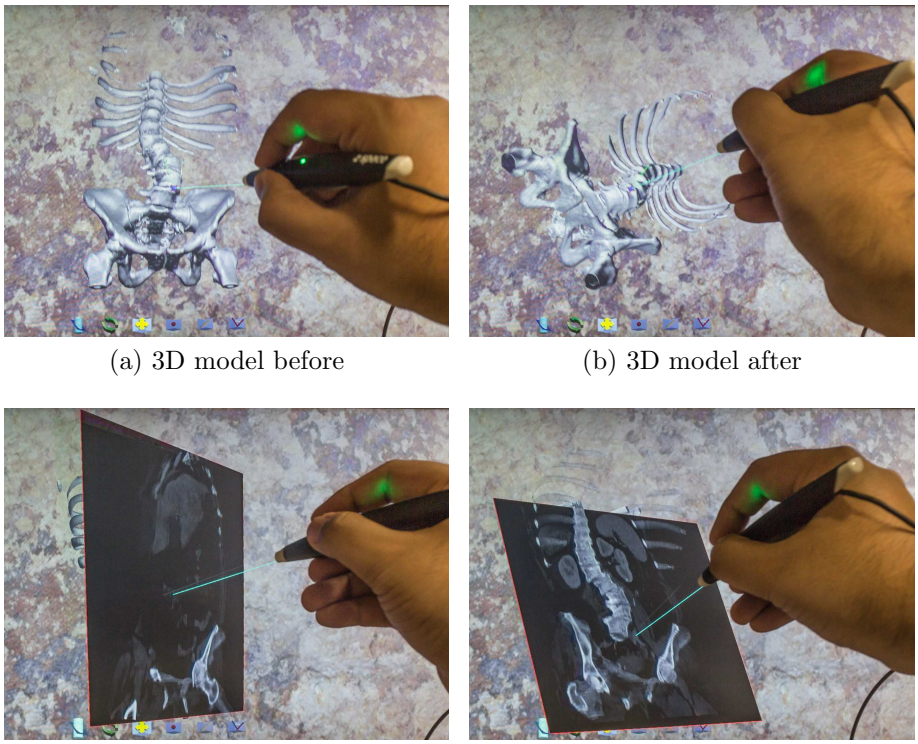
The front button is used to manipulate the objects, the left button is used to translate the objects (while preserving their orientation), and the right button is used to scale the object. The user can change the orientation and scale for the volume mesh and the arbitrary slice only. For other objects, the user is only able to manipulate their position. The manipulation of the model as well as the arbitrary slice can be seen in figure 4.21.

The user can use the reset button to reset the scale, position, and orientation of the volume mesh. The reset button also resets any changes to the position of the light source as well as the arbitrary slice. If the mesh is annotated, or any 3D measurements were performed on the mesh, they retain their relative position within the mesh even after triggering the reset button.

Slice manipulation

The arbitrary slice is hidden by default. The user can show or hide the arbitrary slice using the slice toggle button. Once the slice is visible, the user can manipulate it independent of the volume mesh. The portion of the slice intersecting the mesh shows the volumetric data on the slice as described in section 3.6.

By default, the slice is attached to the mesh. When the user manipulates the mesh, the arbitrary slice maintains its position relative to the mesh. The user can choose to attach or detach the slice from the mesh using the slice attachment button. When it is detached, the volume mesh can be moved independently to the slice, otherwise, the slice always moves relative to the mesh. This can also be



(a) 3D model before (b) 3D model after

(c) slice plane before (d) slice plane after

Figure 4.21: An image of the 3D interface showing model and slice plane manipulation using the 3D stylus input.

used to manipulate the slice to a desired position and orientation.

The three-point slice placement tool allows the user to precisely position and orientate an arbitrary slice within the volume. The user selects the tool using the three-point slice placement button. This allows the user to place three points within the volume creating a triangle. The slice is then aligned with the triangle's plane and centred around the triangle's centroid (see figure 4.22). The slice (normal) is oriented towards the front face of the triangle, which is defined by the clockwise orientation of the vertices. The user can then manipulate the three points individually to position and orient the slice. This tool is particularly useful when the volume consists of well-defined anatomy.

When working with the arbitrary slice in diagnosis, radiologists often scroll the arbitrary slice back and forth to visually inspect the slice plane. To facilitate an easy way to achieve this, the slice oscillation tool was implemented. Once the slice

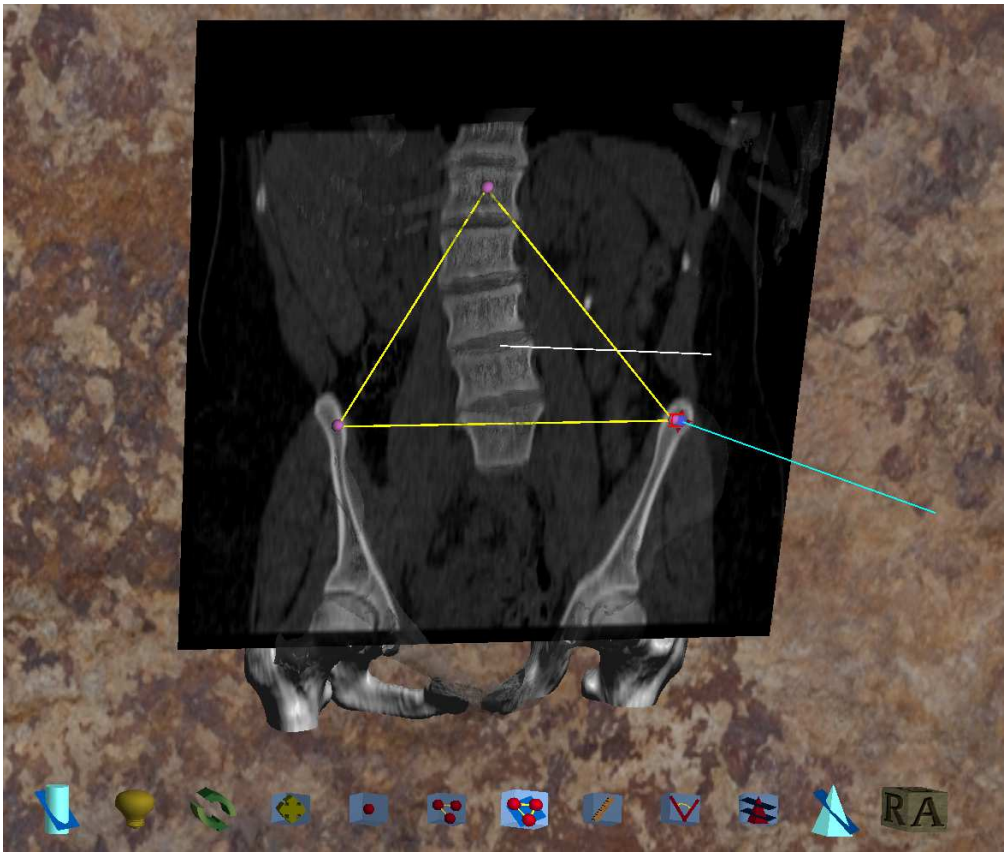
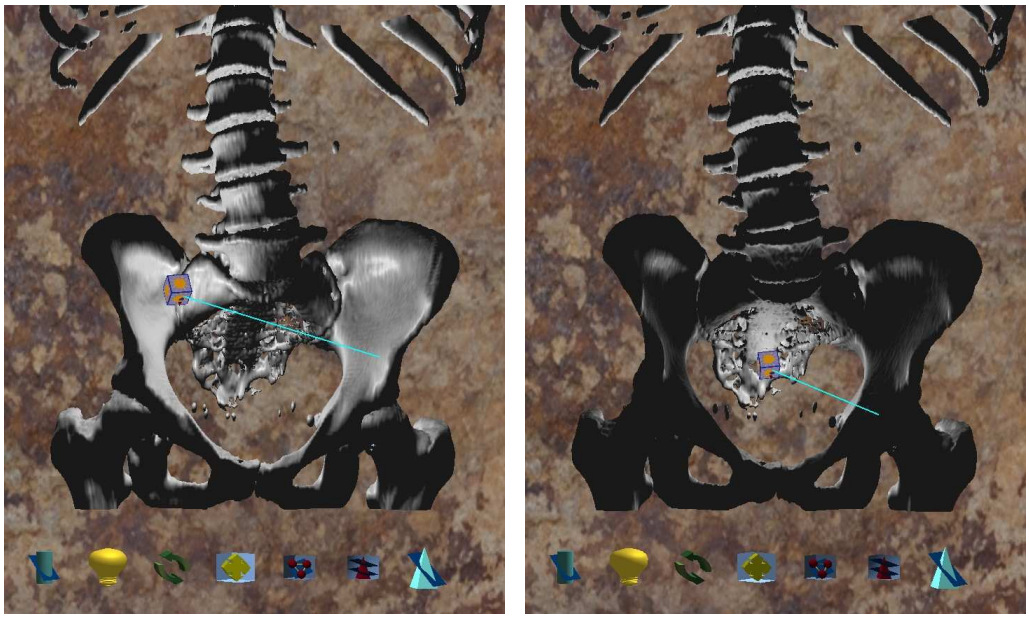


Figure 4.22: Three point slice placement tool.

is manipulated to a desired position and orientation, the user can oscillate the slice (±10 slices) perpendicular to its plane using the slice oscillation button. The oscillation scrolls two slices per second and inverts the direction when it reaches 10 slices. It continues until the user stops the oscillation using the same button or performs any action on the slice such as manipulating it. The slice returns to its original position when the oscillation is stopped.

Light source manipulation

The light source manipulation tool allows the user to pick and move the light source in the 3D scene. The user can show or hide the light source handle using the light toggle button. Once the light handle (a 3D icon representing the position of the light) is visible, the user can change the position of the light source by picking and moving the light handle in the 3D scene (see figure 4.23).



(a) lighting before (b) lighting after

Figure 4.23: Light source manipulation tool showing the change in lighting when the light source is manipulated with the stylus.

The lighting is updated in real-time to reflect the change in the position of the light source. The free motion of the light source allows for easy placement inside the volume.

Point and line annotations

The user can annotate the volume using the point or line annotation tools. These tools can be accessed by the associated line and point annotation mode buttons. The point and line annotations can only be placed inside the volume. When the user manipulates or scales the volume mesh, the point and line annotations maintain their relative position and scale along with the mesh.

The point annotation tool allows the user to define a point (represented by a sphere) inside the volume and annotate it with a label (see figure 4.24). The user can place annotations by pointing to the location in the volume and pressing the right stylus button. A point annotation is placed at the end of the stylus. The user can select and manipulate the annotations within the volume using the front stylus button or delete them using the left stylus button.

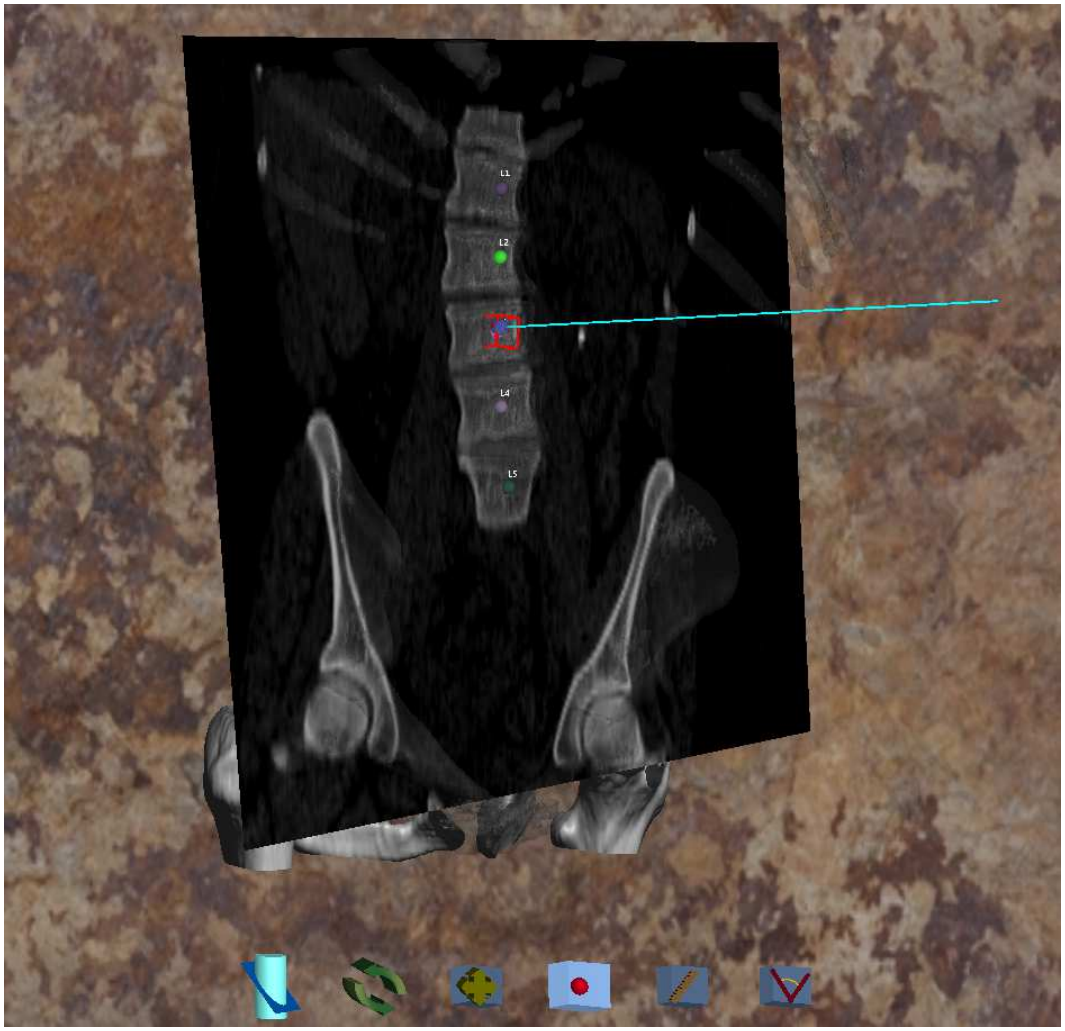


Figure 4.24: Point annotation tool.

Similarly, the user can also define a line segment consisting of a set of points and annotate it. The user clicks the left stylus button to place a new line annotation and add points to the line segment. The user then ends the line segment using the right stylus button. The user can also edit a point on an existing line annotation with the front button. The right button also allows the line annotations to be deleted.

When placing new annotations, the keyboard is used to enter the annotation label. The annotations are stored and synchronized along with the volume. When a user places an annotation in the volume mesh, the annotation can be seen in

the annotations tab in the 2D interface as well as on the associated slices. Alternatively, annotations placed in the 2D view are also visible in their respective locations in the volume mesh.

Line and angle measurements

The line and angle measurement tools are accessible by the associated line and angle measurement mode buttons. The line measurement tool allows the user to draw 3D lines in the volume mesh and measure their length. Line and angle measurements can only be performed inside the volume bounding box. They also maintain their position and scale relative to the volume mesh. The right stylus button is used to define the two endpoints of the line (see figure 4.25).

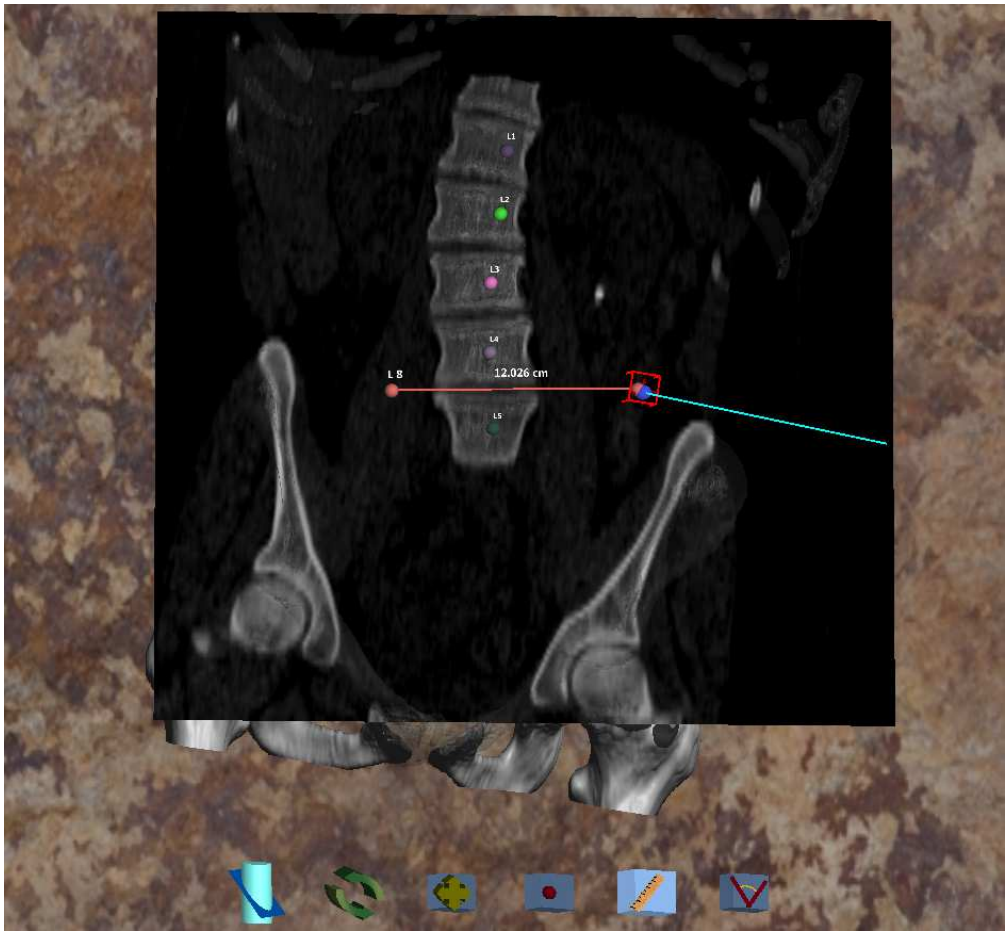


Figure 4.25: Line measurement tool.

Once a line is drawn, the endpoints can be manipulated using the front stylus button and the corresponding length is updated. The left stylus button is used to delete an existing line or cancel the line that is currently being defined.

The angle measurement tool allows the user to select two lines drawn using the line measurement tool and measure the angle between them. This interaction is similar to the 2D angle measurement tool in commercial DICOM viewers as well as tools such as ImageJ [18] which are used by MARS users. The lines are selected by selecting one of their endpoints and pressing the right stylus button. Once two lines are selected, the angle between them is displayed (see figure 4.26).

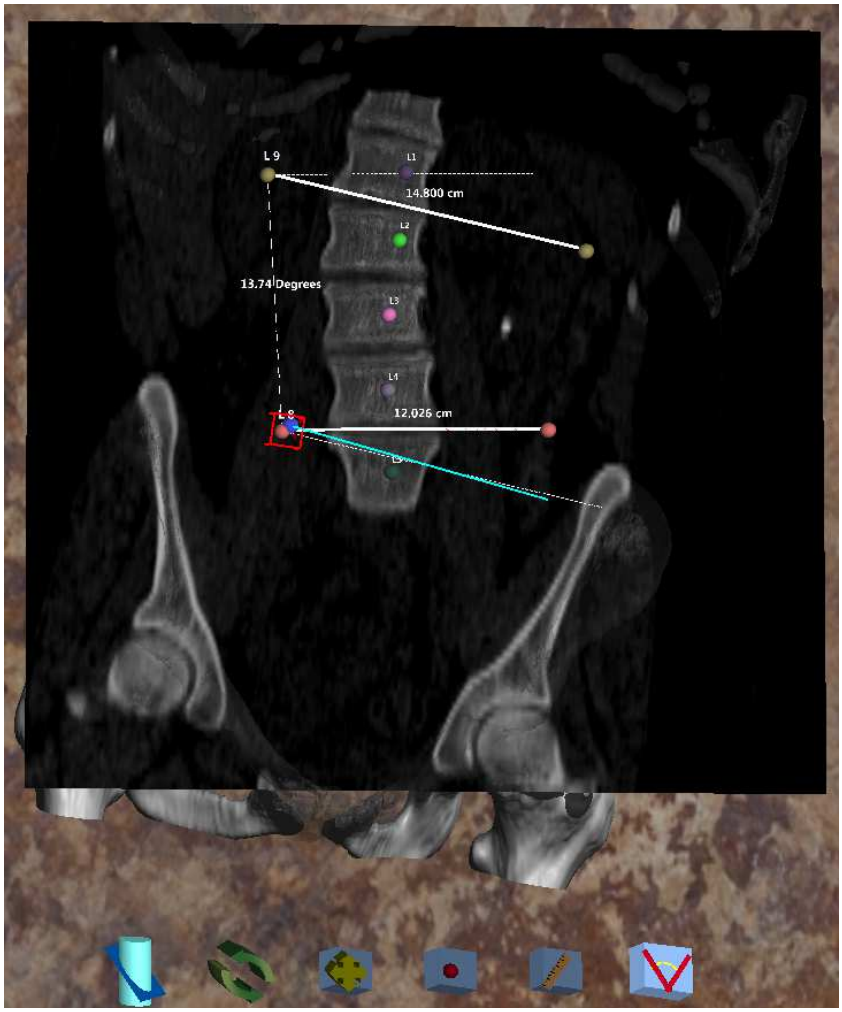


Figure 4.26: Angle measurement tool.

Similarly, the lines can be de-selected using the left stylus button. Once two

lines are selected for angle measurement, the order of their endpoints can be switched using the right stylus button to switch between angles. The endpoints of the selected lines can also be manipulated, and the associated angle is updated in real-time.

4.4 2D/3D Hybrid User Interface

This section describes the 2D/3D hybrid user interface design. It starts by describing the motivation for designing a hybrid interface, followed by an overview of the of the design in section 4.4.2. Later a walkthrough of the interface is explained in section 4.4.3. Finally, the two forms of hybrid interaction (serial and parallel) are discussed in detail in section 4.4.4.

Once the zSpace interface was integrated into MARS Vision, I intended to test the usability of the interface in a CT exploration scenario. The key goal was to compare the effectiveness of 3D stylus input from zSpace with the mouse input for 3D manipulation. Since the MARS user base was limited and not accessible to for running user studies, I chose to explore the area of radiological diagnosis for evaluating the effectiveness of the zSpace interface. The key reasons for this choice included the availability of users for conducting the study, human CT datasets for the study, and the number of well-established diagnosis tasks.

In order to evaluate the 3D stylus input for radiology diagnosis task, it was vital to find a task that required 3D manipulation. Unfortunately, as far as I am aware, there were no radiology diagnosis tasks that only rely on 3D manipulation. Most of the tasks that required 3D manipulation also required some level of 2D manipulation with the anatomical slice. This was the initial source of inspiration for exploring the literature around hybrid interfaces combining 2D and 3D interaction.

4.4.1 Motivation for Hybrid User Interface

Most radiology tasks involve both 2D and 3D image/object interaction, so instead of doing both with a 2D interface, it makes sense to explore a hybrid interface option. Some VR systems also rely on some level of 2D user interaction. Hence, researchers started finding ways to combine 2D and 3D interaction to form hybrid user interfaces. Since early VR hardware had limited resolution, early hybrid in-

terfaces focused on providing means for completing high-resolution tasks in virtual environments [95].

The most common approach has been integrating hand-held and touch-screen devices into virtual environments for 2D interaction. An interface projecting a virtual computer that appeared as a hand held display for providing input within a virtual environment was developed by Angus et al. [96]. A free standing 6 DOF device with a screen attached on top was designed by Hachet et al. [97]. More recently, an interface combining a tablet with a head-mounted display (HMD) based virtual environment was proposed by Wang et al. [98].

Other researchers focused on providing hybrid input in virtual environments. The Virtual Tricoder by Wloka and Greenfield duplicated a 6 DOF input device in the virtual environment in a one to one fashion to help assist the user in establishing a relationship between the tactile and visual interface [99]. A Pinch Glove based hybrid interface using a 2D/3D cursor for object modelling was proposed by Coninx et al. [100]. A Pick and Drop approach using a pen to move objects across a single or multiple computers was proposed by Rekimoto et al. [101]. An interface for desktop manipulation with tangible user interfaces was proposed by Ullmer and Ishii [102].

Some developments involved multi-display setups. For example, the hybrid interface for visual data exploration combining the 3D display (with glove input) and a tablet PC, which was proposed by Baumgärtner et al. [103]. Similarly, in the medical domain, another hybrid interface for manipulation of volumetric medical data for liver segmentation refinement was developed by Bornik et al [104]. This interface combined a tablet PC and an immersive VR environment. They designed a stylus for 2D interaction on the tablet PC that also acted as a 3D wand for the VR environment. The stylus along with the tablet PC was used for precise 2D input and system controls. The 3D interaction of the stylus in VR was used for 3D manipulation. They found that this approach was more efficient compared to a 2D only interface.

I extend their approach by combining the zSpace with a traditional 2D system for my hybrid interface. The key difference is that while they combined the 2D interaction on a tablet PC with an immersive VR environment, I combine a standard 2D desktop PC with a desktop/fish-tank VR system (the zSpace).

A 3D volume navigation tool for diagnostic radiology, that used a 3D mouse to

provide both 2D as well as 3D input for exploring volumetric data, was developed by Teistler et al. [105]. The ability to effectively navigate through 3D volumetric data could improve their understanding of complex anatomical structures leading to more accurate diagnosis. A tool for post-mortem CT visualisation using a 3D gaming controller for interaction along with an active stereo screen was proposed by Teistler et al. [106]. It showed that using 6 DOF input can be very intuitive compared to 2D input, especially for the layperson.

Another approach from Graves et al. [107], used a fixed 6 DOF controller to reformat a slice in 3D data and found that radiologists were faster with their approach, compared to mouse input from the standard radiology workstation. A key limitation of their study was that the task involved achieving a specific anatomic orientation and very similar images could be achieved with slightly different oblique planes. This highlighted the need to choose a diagnosis task that can more accurately predict the effectiveness of an interface.

The literature shows that a number of researchers explored the benefits of hybrid interfaces for 3D object manipulation. However, as far as I am aware, there are very few medical interfaces that use hybrid input, and none that combine 2D and 3D screen viewing with stylus interaction in diagnostic radiology. Hence, I was motivated to design a hybrid interface for MARS Vision, combining the zSpace desktop VR system with the traditional 2D system with mouse input to form a 2D/3D hybrid user interface.

Another key motivation for combining a traditional system with the zSpace system was an interface from Draken and Durost, which combined a tablet PC and an immersive 3D VR environment (three-sided CAVE system) [108]. They used a stylus that worked on the tablet for 2D interaction and it could also be used for 3D interaction by pointing at objects in the 3D VR system. They also present a study that compared using 2D interaction for 2D tasks and 3D interaction for 3D tasks, which found that an appropriate mix of 2D and 3D tasks can yield better overall performance. They stated that mixing the two forms of interaction prevented loss of performance in 2D tasks in favour of 3D and vice versa.

4.4.2 Interface Design

The hybrid system combines 2D and 3D interface components for exploring CT data (see Fig. 4.27). The 3D component consists of the zSpace 3D stylus input and the zSpace stereoscopic display with head tracking, discussed in the previous section 4.3.

The 2D component consists of mouse input and a 2D display. The interface for the 2D component is based on the MARS Vision interface described previously in section 2.2.2. This default setup is different to that of the previous version of MARS Vision. The old layout was a 2×2 grid of views with 1 3D view and 3 anatomical views. In the hybrid case, the 2×2 grid is retained except that instead of the 3D view, you now have the arbitrary 2D slice view. The 3D is also present on the zSpace monitor as a fifth view.

The hybrid system displays anatomical objects in 3D along with a 2D slicing plane on the zSpace stereoscopic display. The anatomical object, as well as the slice plane, can be manipulated in 3D using the 3D stylus input as discussed in section 4.3.4. The slice plane is also displayed synchronously on the 2D display.

This concept of a synchronized slice being displayed in both 2D and 3D views is similar to the approach used by Teistler et al. [105]. They displayed both the 2D and 3D views side by side on the same display with the 3D view showing the 2D slice along with the 3D model. The major differences between their interface and my hybrid UI were that they did not use a stereoscopic display, their slice plane clipped the 3D volume and they used a standard mouse with a 6 DOF tracker for input.

Measurements and annotations can be performed using either the 2D or the 3D components of the hybrid interface. Both the annotations and the measurements are also synchronized between 2D and 3D views. Displaying 2D measurements in the 3D view is similar to the approach used by Preim et al. [29] where they draw 2D measurements snapped to the corresponding 2D slice. They also flatten some 3D measurements such as spheres into circles for display on the corresponding 2D slices, which is not supported in the hybrid UI.

This enables experienced users to utilize familiar 2D views to explore information within the slice while using 3D stylus input for 3D manipulation of the slice. It also allows less experienced users to utilize the 3D display, to gain an overview

Figure 4.27: An image of the hybrid interface showing the 2D display at the top, followed by the zSpace display at the bottom. The image also shows mouse, keyboard and the 3D stylus input.

of the anatomy, which in turn helps them identify important slices and explore the information within.

4.4.3 Interface walkthrough

When a CT dataset is loaded by the user, the 2D interface displays the three anatomical planes along with the arbitrary plane in a 2 by 2 grid layout. The arbitrary plane is initially a clone of one of the three anatomical planes.

A 3D mesh is extracted from the CT dataset (see section 3.4) and used as a 3D model representation of the CT dataset. A new mesh is extracted each time it requires a change and not every frame.

Once the default 2D and 3D views are set up, the user can interact with the 2D interface by selecting and scrolling through any anatomical plane using the mouse. If the 3D plane is enabled in the 3D interface, its position and orientation on the object (volume mesh) are synchronized to match the selected 2D plane.

The plane displays the CT numbers for a CT dataset, the linear attenuation coefficients for MARS attenuation volumes, and material densities for MARS material volumes. This plane along with the volume model provides a rapid assessment of the arbitrary plane's position in the model.

The user can synchronize the arbitrary plane orientation and position to any of the three orthogonal planes by selecting the orthogonal plane by left clicking on it while holding down the [control] key. The user can rotate the arbitrary plane to a desired orientation using the mouse in the 2D interface. The slice is always rotated about its own centre point.

When the arbitrary slice is selected, its guidelines are displayed on the other orthogonal views. These guidelines can also be used to manipulate the position and orientation of the arbitrary slice. These actions are synchronized with the plane in the 3D interface.

The 2D interaction method for slice views in the hybrid UI is equivalent to that used in Inteleviewer. This is important because Inteleviewer is the commercial DICOM tool used for human diagnosis at the Christchurch Hospital.

Annotations are placed by right-clicking on the slice, while the annotation mode is selected. Lines can be drawn by right-clicking on the slice to start the line and dragging to the desired endpoint, while the line measurement mode is selected.

The line can then be moved, or modified by moving its endpoints.

The angle measurement mode allows the selection of two lines to measure the angle. If only two lines are drawn and the user switches to the angle measurement mode, these lines are automatically selected and the angle between them is displayed, as shown in figure 4.28. Annotations and measurements on the slice view, are also displayed in the 3D interface.

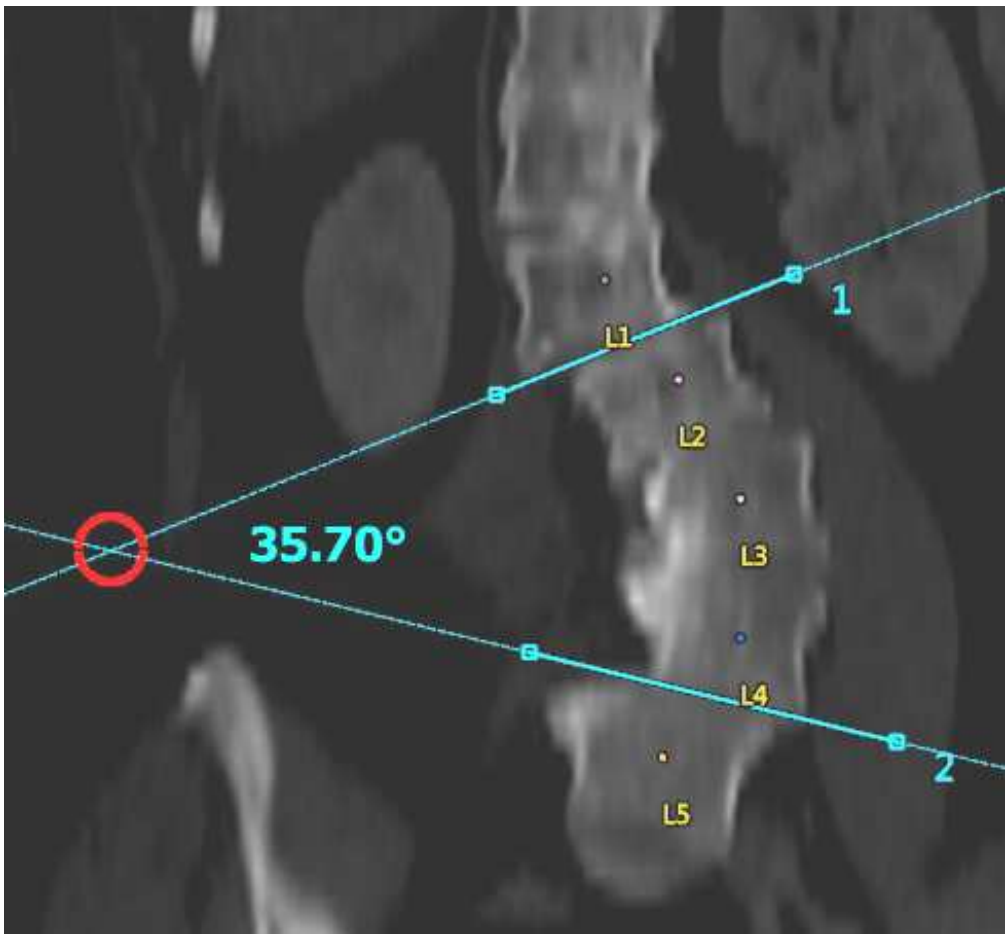


Figure 4.28: 2D interface demonstrating the annotation and measurement tools used for a scoliosis angle measurement task on the arbitrary slice.

The 3D model, as well as the arbitrary slice plane, can be manipulated using the zSpace stylus (see section 4.3.5). For the hybrid interface, only the direct manipulation method is used, which maps the stylus motion to the objects in a one to one fashion (see section 4.3.4). By default, the plane maintains its position relative to the 3D model when the model is manipulated.

Changes made to the plane orientation are always updated in the 2D interface. This can be seen in Fig. 4.27, where the 3D plane and the arbitrary plane (top right) are showing the same information. The user can also perform other functions, such as placing annotations (see section 4.3.5) or performing measurements (see section 4.3.5), in the 3D interface, as shown in Fig. 4.29.

Figure 4.29: An image of the 3D interface showing annotations and measurements tools used for a scoliosis angle measurement task.

At any stage, the annotations or measurements can be picked and moved using the front stylus button. They maintain their relative position to the 3D model when the model is manipulated. They also scale with the model.

If the arbitrary plane is enabled while placing annotations or measurements, they will automatically snap to the slice plane, if they are within the distance of 0.5 cm from it. This makes it easy for the user to place measurements on a slice plane, using the 3D stylus input.

Annotations such as 3D points are shown in the 2D interface on the corresponding slice. However, the 3D measurements such as 3D lines are not shown in the 2D interface. The user can choose to interact with the 2D or 3D interface, or a combination of both in order to simply explore a dataset or perform diagnosis.

4.4.4 Hybrid Interaction

Hybrid interaction can be performed in two ways, serial and parallel. In the serial approach, the tasks are performed using the 2D and 3D components independently in a sequential manner. The diagnosis task can be divided into multiple steps, each of which can be performed using 2D or 3D components alone. Since both the components are synchronized with each other, the user can proceed with either interface for the next step. For example, the user can position and orient the arbitrary slice using the stylus and later use the mouse to perform 2D measurements on it.

In the parallel approach, both the 2D and 3D components can be used together at the same time to perform a single step in the diagnosis task. For example, a user can use the stylus to manipulate the arbitrary plane, while looking at the slice content on the 2D interface. The user can also choose to scroll through a slice using the mouse while looking at the 3D model to understand its relative position. Ambidextrous users can use the mouse and stylus together to manipulate the slice and the 3D model simultaneously.

Both approaches are supported in the hybrid system, however, only the serial approach was tested in the evaluation (see Chapter 5), as it required considerably less training for users who have prior experience with the 2D component. Since the serial approach uses only mouse or stylus input at any given time, it can be easier, and faster, to learn the system.

I believe that hybrid interaction can have considerable advantages. The 2D component can be used for high-precision tasks such as measuring line lengths, angles between lines, or marking 2D ROIs. Data inspection can also be performed

in higher resolution since it is possible to view the slice information with a one to one mapping between the slice data and pixels. This is not ideal using the 3D component since slice information is textured on a 3D plane, which is often in a non-optimum orientation when viewing it in 3D. Additionally, the circular polarization of the stereoscopic screen also reduces image brightness making it harder to view the slice information in the 3D view.

The 3D component can be used to gain a quick overview of the anatomy by manipulating the displayed anatomical object. It can also be used for tasks such as slice plane orientation, and 3D line and 3D ROI measurements. The speed/accuracy trade-off will depend heavily on the type of exploration or diagnosis task. The key factors would be the flow of actions in performing a particular task, the level of synchronization and the user's ability to shift focus between the 2D and 3D components.

4.5 Summary

This chapter discussed the development of a 2D/3D hybrid user interface that aims to provide effective 3D manipulation for the exploration of MARS datasets. Previous research is presented, which shows that 3D input devices can outperform the mouse for 3D manipulation. This motivated the exploration of various 3D input devices in literature. The SpaceMouse interface is implemented in MARS Vision for manipulating the volume model and the arbitrary slice. Feedback from MARS users suggested that the SpaceMouse required substantial training before it could be used for manipulation. This highlighted the need for a more natural interface that was easy to learn.

Later, the zSpace interface was implemented for MARS Vision. The feedback from MARS users suggested that it was intuitive and easy to learn. In order to test the effectiveness and usability of the zSpace interface, the area of diagnostic radiology was chosen due to the limited availability of MARS users. Most radiology tasks still required some level of 2D manipulation. This combined with the literature, which suggested that combining 2D and 3D input can be more efficient, inspired the design of the hybrid user interface combining the zSpace interface with the standard 2D system. Finally, a walkthrough of the interface for a typical radiology task is presented and the hybrid interaction is discussed in detail. An

evaluation of the hybrid interface is presented in the next chapter.

Chapter V

Evaluation

This chapter discusses the evaluation of the interactive hybrid interface. The goal of the evaluation is presented, followed by exploring the radiology workflow. Later, the methodology is discussed in detail. This includes the participants, the task and datasets, the experimental process, and the measurements. Finally, the statistical analysis results of the evaluation are presented, followed by a detailed discussion of the results. The results from this chapter were published in a journal article [109], where I was the first author.

The goal of this evaluation was to determine the effectiveness of the hybrid interface for radiological diagnosis compared to a more traditional 2D interface. This was achieved by comparing traditional 2D slice manipulation tools with the hybrid interaction tools for completing a representative radiological diagnosis task that required 3D manipulation. The hybrid interface is a combination of 3D and 2D components, so a 3D only interface was also included to eliminate the possibility of the observed difference being due to the 3D component alone. To summarize, three conditions were evaluated: 2D, 3D, and hybrid.

5.1 Exploring radiology workflow

Most radiology diagnosis tasks are performed by exploring one or more of the three anatomical planes: sagittal, coronal, and transverse planes. The procedure for diagnostic radiology widely varies depending on the task. Conventional radiology software focuses on exploring 2D slices for diagnosis. Radiologists receive extensive training on how to mentally visualize 3D anatomical objects from these slices. It is only after acquiring a lot of experience, they are able to do this task well [110].

Creating an accurate 3D mental model helps them associate the location of each slice with its position inside the anatomical object. They find and examine

the interesting slices, identify anomalies or features within these slices, and report their findings. Although it is possible to render 3D anatomical objects from 2D slice data (for example volume rendering [111]), radiologists seldom rely on it. They find it easier and faster to diagnose using the 2D slices alone.

5.2 Methodology

The key research question was to determine if the hybrid interface provides any improvements over the existing 2D interface in terms of task performance and accuracy. Furthermore, I wanted to study how the user's prior experience with the 2D interface, influenced their task performance with the hybrid interface. Hence, the experiment was set up as a mixed *within/between* study in which the interface (2D, 3D, hybrid) was the within factor, and experience (student, resident) was the between factor. A human ethics approval for the user study was obtained from University of Canterbury Human Ethics Committee (Ref# HEC 2016/35/LR-PS) [112] (see Appendix A.1).

5.2.1 Participants

For the evaluation, I chose two groups of participants: fourth-year medical students and radiology residents (physicians training to become radiologists). Fourth-year medical students have the required medical background necessary to perform some diagnostic tasks but have little to no experience using any diagnosis tools. Residents have at least one to two years experience using a 2D only interface, using it daily for various diagnosis tasks.

Since the hybrid interface uses a stereoscopic 3D display, only participants with binocular or stereo vision were chosen. Since 3D stylus interaction was also involved, we only chose participants who had no hand disabilities or pre-existing conditions that would prevent them from using the stylus. A simple finger test [113] for depth perception was used to test binocular vision. There were a total of 31 participants for the evaluation. The student group consisted of 21 participants, while the resident group consisted of 10 participants.

The student group had 11 male and 10 female participants with an average age of 22. The resident group had 6 male and 4 female participants with an average age of 32. Although I did not aim for equal participants from each gender,

the distribution was approximately even between genders. I planned to use age and gender as factors during the statistical analysis to see if they had any strong correlations with task performance.

5.2.2 Task

After consultation with experienced radiologists, a real diagnosis task was chosen, namely obtaining a scoliosis angle measurement from abdominal CT scans. Scoliosis is a medical condition where a person's spine curves sideways which can cause respiratory issues in severe stages. The angle of this curve is measured by radiologists which helps determines the severity and treatment. Initially this angle (also known as Cobb angle) was measured on coronal radiographs [114], however, it is currently measured using CT.

This task consists of three simple steps: correcting the coronal slice orientation, annotating the spinal column, and measuring the scoliosis angle. Hence, the task could be performed by anyone with a basic knowledge of human anatomy. This would largely reduce the possibility of any participant being affected by the task difficulty.

Only the abdominal CT scans where the patient's spinal column was not parallel to the coronal plane were chosen, as this was a very common scenario for such scans. Hence, it was guaranteed that the diagnosis task involved 3D manipulation of the coronal plane, in order to correct its orientation. There were three similar, but different, datasets used for the experiment. The order of the conditions and the use of datasets were both counterbalanced using a balanced Latin Square [115].

5.2.3 Process

Each participant first received a five-minute interface demonstration from an instructor followed by another five minute practice period. During the practice period, the participant was allowed to perform the diagnosis task with a practice dataset and ask any questions regarding the task or interface.

Later the participant performed the actual diagnosis task on a different dataset to the one used for practice. During this period, the participants were not allowed to ask questions, unless they were absolutely stuck. They were encouraged to rely on the provided reference cards, and the application help menu, to guide them

through the task. This was done to minimize interference, and to ensure that the same amount of help was available to every participant, during the actual task.

The scoliosis task consisted of three steps. The first step was is to adjust the orientation of the arbitrary 2D (coronal) slice to optimally display the spine by correcting for the tilt. The second step was to identify and annotate the vertebrae, on the arbitrary 2D slice. Finally, the user drew two lines between two sets of vertebral discs and measured the scoliosis angle between them. This procedure was repeated for each interface condition.

An example measurement of the scoliosis angle using the 2D interface can be seen in figure 4.28 and using the 3D interface in figure 4.29. Keyboard input was used to name the annotations in all conditions. The 2D condition involved 2D mouse input while the 3D condition used the 3D stylus as the input. The hybrid condition was a mixture of both, utilizing the 3D stylus input for the first step and mouse input for the rest of the task.

After each diagnosis task, participants were required to complete two written questionnaires. See section 5.2.4 for the details. At the end of the experiment, they were asked to comment on their experience in a semi-structured interview.

To better control the environment of the task, the window/level settings to make the bones clearly visible within the slice were pre-set. This guaranteed the same initial conditions for the second step. In addition, the task datasets that required the same set of vertebral discs to be measured for the scoliosis angle were chosen. This was communicated to the participants prior to starting each diagnosis task. Participants were also given reference cards showing a labelled diagram of a healthy spinal column along with a guide to measuring the scoliosis angle and interface controls for all conditions, as shown in Appendix B.

5.2.4 Measures

The study focused on two types of measures: performance and usability. The performance measures consisted of completion time and accuracy. These could be used to quantify the effectiveness of the hybrid system. The usability measure consists of two standard usability questionnaires: system usability scale (SUS) and NASA task load index (TLX).

Performance measurements

For each task, a snapshot of the labelled vertebrae, the completion time and resulting scoliosis angle were recorded. The completion time was recorded in *seconds* and the scoliosis angle was recorded in *degrees*. The snapshot was used to verify the correct annotation of the spinal column. If the spinal column was incorrectly labelled, any resultant angle measurement would be considered incorrect. The labelled snapshot was used to ensure the correct comparison of scoliosis angles for measuring accuracy.

The measured angle was later compared with a reference solution for the same dataset, to establish an accuracy measure for each diagnosis task. A senior radiologist provided a set of reference measurements for all datasets used in the experiment. This reference measure was obtained by taking an average of three measurements performed by the senior radiologist, where the variation between these three measurements was 0.5°. This accuracy measure was the absolute error in the scoliosis angle, which was used for comparison.

Usability measurements

In order to assess the usability of the interface, the System Usability Scale (SUS) questionnaire [116] was chosen because it is a well-validated measure used by thousands of studies. Research shows that a mean SUS score greater than 70 corresponds to an above average adjective rating and a score above 80 is considered very good [117]. Note that although the SUS score is on a scale of 0 to 100 it should not be interpreted as a percentage that directly depicts the usability of the interface [118].

The physical and mental task load were also measured using the NASA TLX [119] questionnaires. Since the evaluation task was a relatively short task, only the physical and mental task load scales from the NASA TLX questionnaire were used for the study.

The 2D and hybrid conditions involved a fixed number of mode changes. However, the 3D condition involved multiple mode changes for performing the diagnosis task. While performing the task in the 3D condition, the user was required to change to slice manipulation mode, point annotation mode, line measurement mode, and angle measurement mode. A minimum number of four mode changes

were required. Hence, an additional measure of the number of mode changes in the 3D condition was also recorded for each participant. An additional number of mode changes would be required if the user made mistakes. Hence this measure was an indication of how easy it was to learn the 3D user interface.

Data analysis strategy

I wanted to compare the mean differences between groups split on two independent variables in the evaluation: *interface* and *experience*. The interface represents the 2D, 3D, and hybrid conditions. The experience represents the student and resident groups. Hence, a *two-way mixed analysis of variance (ANOVA)* was chosen, to understand any interaction between the two independent variables on the dependent variables.

The independent variables consist of a between-subjects factor and a within-subjects factor. The between-subjects factor was the *experience* and the within-subjects factor was the *interface*. The purpose was to find if statistically significant interaction effects exist between the interface and experience. If significant interaction exists, the simple main effects of the interface for each experience group and vice-versa, as well as the main effects of the interface and experience would be reported independently. If no statistically significant interaction effect exists, only the main effects would be reported.

Each participant performed the diagnosis task under all three interface conditions. Hence, for investigating the simple main effects, a *one-way repeated measures ANOVA* was chosen to analyse the difference in effects of the interface on each experience group. Additionally, the Bonferroni post-hoc test was used to adjust the confidence intervals for significant interaction effects. To investigate the difference in effects between experience groups on each interface condition, I chose to use the *independent samples t-test*.

5.3 Results

The key question that the study aimed to answer was whether the hybrid interface was more effective than existing 2D interfaces, or the 3D only interface, for a scoliosis diagnosis task. To answer this question, a two-way mixed ANOVA with two factors was conducted: interface (2D, 3D, hybrid) and experience (student,

resident) on the resulting experimental data with completion time, absolute error, SUS score, physical task load and mental task load as the dependent variables.

To identify significant outliers, the descriptive statistics for task completion time were considered. There were two significant outliers in the data, as assessed by inspection of a boxplot for values greater than 3 box-lengths from the edge of the box (see figure 5.1).

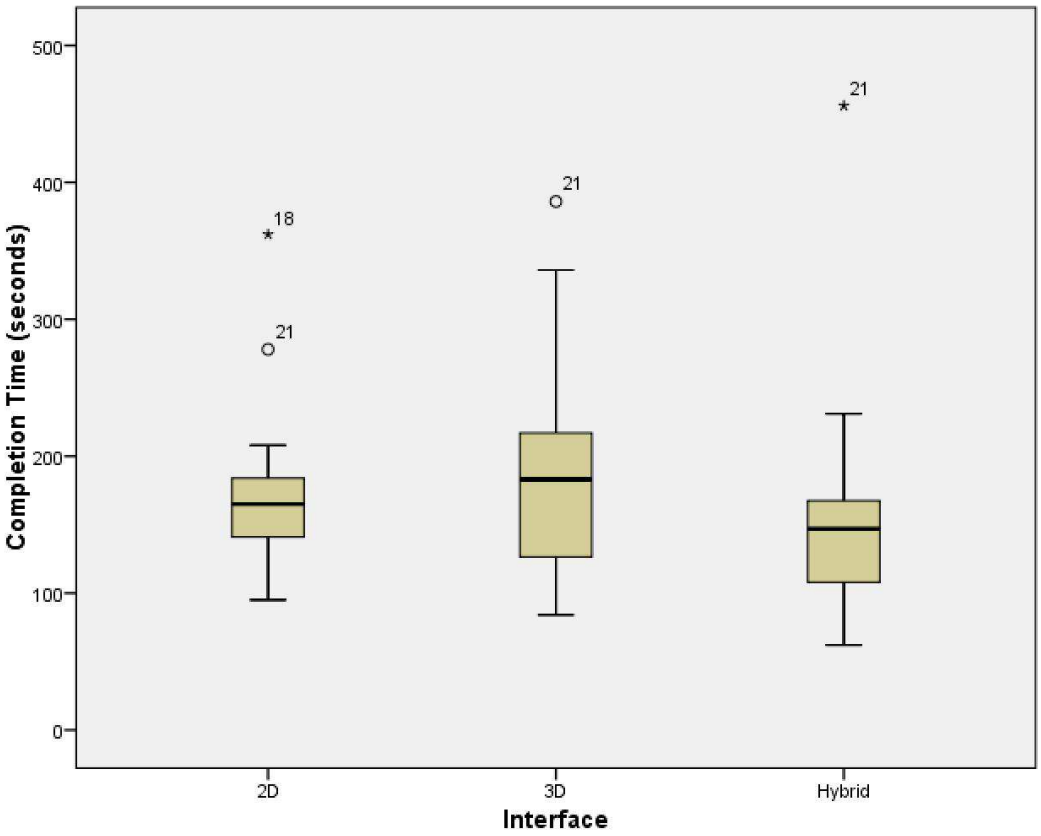


Figure 5.1: Boxplot for Completion Time. The number of participants ($N = 31$). The box represents the inter-quartile range (IQR) of the middle 50% of the participants. The horizontal line in the box represents the median value for completion time. The whiskers extend to the lowest and highest values for completion time within $1.5 \times IQR$. The circles represent the outliers that lie beyond $1.5 \times IQR$ but within $3.0 \times IQR$. Data points beyond $3.0 \times IQR$ are represented by a star symbol. The participant numbers are shown next to the outliers.

For task completion time, the values greater than 3 box lengths included participant 18 for the 2D interface ($time = 362$, $Mean = 168.58$, $SD = 50.06$)

and participant 21 for the hybrid interface ($time = 456$, $Mean = 150.77$, 70.90). Hence, participants 18 and 21 were labelled as significant outliers.

To determine whether these outliers are statistical outliers, Grubb's test [120] was chosen. The studentized values for task completion time for participant 18 for the 2D interface ($T_{18} = 3.86$) and participant 21 for the hybrid interface ($T_{21} = 4.32$) were computed using the equation 5.1, where T_i is the studentized value, x_i is the value for i^{th} participant, $\bar{x}$ is the mean value, and s is the standard deviation.

$$T_i = \frac{(x_i - \bar{x})}{s} \tag{5.1}$$

These studentized values were then compared to the critical value ($T_{critical} = 2.644$) from Grubb's table [120] for sample size ($n = 21$) at the 95% confidence interval. Both the identified outliers had studentized values well above Grubb's critical value ($T_i > T_{critical}$). Hence, both participants 18 and 21 were confirmed as statistical outliers. Upon reviewing the observation notes and videos for these participants, a hardware malfunction was identified as the cause. Thus, both these outliers were completely removed from the analysis.

The data was normally distributed, as assessed by Shapiro-Wilks test of normality ($p > 0.05$). There was a homogeneity of variance ($p > 0.05$) and covariances ($p > 0.05$), as assessed by Levenes test of homogeneity of variances and Box's M test, respectively.

Mauchlys test of sphericity indicated that the assumption of sphericity was met for the two-way interaction for: completion time ($\chi^2(2) = 1.238$, $p = 0.538$), absolute error ($\chi^2(2) = 1.324$, $p = 0.516$), physical task load ($\chi^2(2) = 0.262$, $p = 0.877$), and mental task load ($\chi^2(2) = 3.394$, $p = 0.183$), but was violated for the SUS score ($\chi^2(2) = 20.934$, $p < 0.0005$). The Greenhouse-Geisser correction was used for the SUS score ($\epsilon < 0.75$) [121].

There was a statistically significant interaction between the interface and the experience on the completion time ($F(2, 54) = 37.835$, $p < 0.0005$, partial $\eta^2 = 0.584$), absolute error ($F(2, 54) = 4.416$, $p = 0.017$, partial $\eta^2 = 0.141$), SUS score ($F(1.288, 34.772) = 13.604$, $p < 0.0005$, partial $\eta^2 = 0.335$, $\epsilon = 0.644$), and physical task load ($F(2, 54) = 5.564$, $p = 0.006$, partial $\eta^2 = 0.171$). This was not the case for the mental task load ($F(2, 54) = 0.053$, $p = 0.948$, partial $\eta^2 = 0.002$).

For the mental task load, since the interaction was not statistically signifi-

cant, the main effects were investigated, but there was no statistically significant difference between the interfaces ($p = 0.089$), or experience ($p = 0.161$).

5.3.1 Simple Main Effects

Since statistically significant interaction implies that the impact of the interface is dependent on the group, the simple main effects of the interface for each group individually and vice-versa were investigated. For the remainder of this section, the results are reported in two parts.

The first part consists of the effects of each *interface* on *experience groups* for the dependent variables. This is done by first splitting the data into two experience groups: students and residents, followed by a one-way repeated measures ANOVA on each group with the interface as the repeating factor. The mean (M), standard error (SE) and p-value are reported for each combination of interfaces.

The second part consists of the effects of each *experience group* on the *interfaces* for the dependent variables. This is done by running the independent samples t-test on the dependent variables for all interfaces with *experience* as the grouping variable with two defined groups: students and residents. The results are reported as mean ± standard error ($M \pm SE$), followed by the t-value and p-value for the mean difference. The effect size (d) for the t-test is also computed and reported using Cohen's d (see equation 5.2).

$$d = \frac{|M_1 - M_2|}{S_{pooled}} \tag{5.2}$$

Where d represents Cohen's d, M_1 and M_2 represent sample means for the two measures in the independent samples t-test, and S_{pooled} is the pooled standard deviation given by equation 5.3.

$$S_{pooled} = \sqrt{\frac{S_1^2(n_1 - 1) + S_2^2(n_2 - 1)}{n_1 + n_2 - 2}} \tag{5.3}$$

Where S_{pooled} is the pooled standard deviation, S_1 and S_2 are the two standard deviations, and n_1 and n_2 are the two sample sizes for the two measures in the independent samples t-test.

Effects of interface and experience on completion time

The mean completion time for both groups of subjects is shown in figure 5.2. For the student group, the task completion time was statistically significantly higher while using: the 2D interface compared to the 3D interface ($M = 24.21$, $SE = 7.91\,seconds$, $p = 0.020$), the 2D interface compared to the hybrid interface ($M = 51.26$, $SE = 8.77\,seconds$, $p < 0.0005$), and the 3D interface compared to the hybrid interface ($M = 27.05$, $SE = 9.92\,seconds$, $p = 0.042$).

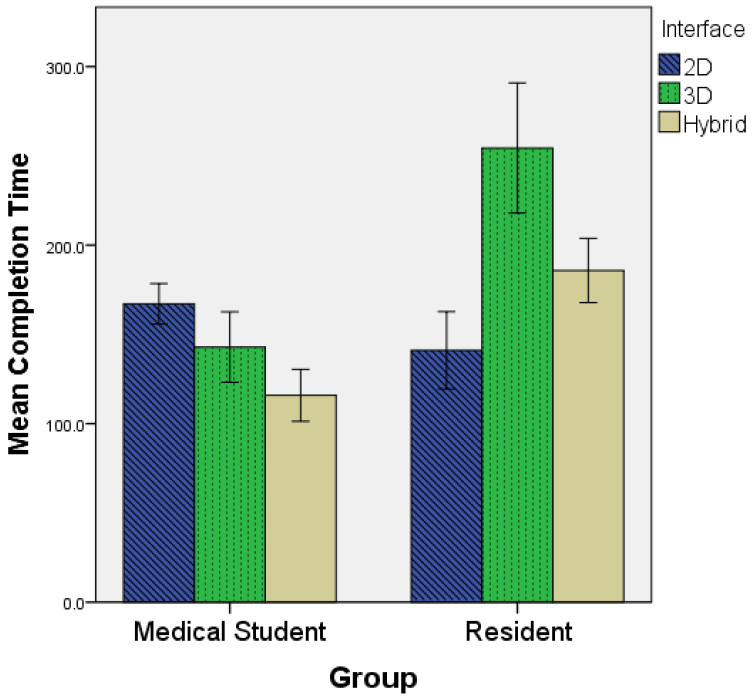


Figure 5.2: Mean completion time

For the residents, the task completion time was statistically significantly lower while using: the 2D interface compared to the 3D interface ($M = 113.3$, $SE = 16.27\,seconds$, $p < 0.0005$) and the 2D interface compared to the hybrid interface ($M = 44.7$, $SE = 11.67\,seconds$, $p = 0.012$). The task completion time was statistically significantly higher while in the 3D interface compared to the hybrid interface ($M = 68.6$, $SE = 15.68\,seconds$, $p = 0.005$).

The completion time of students when compared to residents was statistically significantly higher while using the 2D interface ($26 \pm 10.15\,seconds$, $t(27) = 2.563$,

$p = 0.016$, $d = 1.0$). The completion time was statistically significantly lower while using: the 3D interface ($111.51 \pm 17.38\,seconds$, $t(27) = 6.415$, $p < 0.0005$, $d = 2.51$) and the hybrid interface ($69.96 \pm 11.19\,seconds$, $t(27) = 6.255$, $p < 0.0005$, $d = 2.44$).

Effects of the interface and experience on absolute error

The mean absolute error for both groups of subjects is shown in figure 5.3. For the student group, the absolute error was statistically significantly higher while using: the 2D interface compared to the 3D interface ($M = 4.29$, $SE = 0.91°$, $p = 0.001$) and the 2D interface compared to the hybrid interface ($M = 4.789$, $SE = 0.86°$, $p < 0.0005$). The absolute error was not statistically significantly different between using the 3D and hybrid interfaces ($M = 0.502$, $SE = 0.743°$, $p = 1.0$).

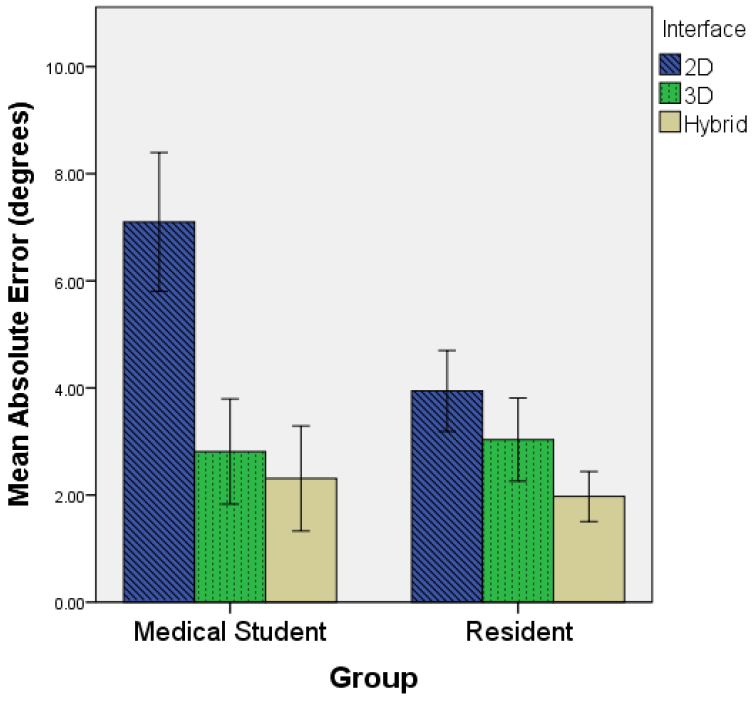


Figure 5.3: Mean absolute error

For the resident group, the absolute error was statistically significantly higher while using the 2D interface compared to the hybrid interface ($M = 1.97$, $SE =$

0.50°, $p = 0.01$). The absolute error was not statistically significantly different between the 2D interface compared to the 3D interface ($M = 0.91$, $SE = 0.47°$, $p = 0.257$) and the 3D interface compared to the hybrid interface ($M = 1.06$, $SE = 0.44°$, $p = 0.117$).

The absolute error of students when compared to residents was statistically significantly higher while using the 2D interface ($3.15 \pm 0.70°$, $t(25.7) = 4.494$, $p = 0.001$, $d = 1.38$). There was no statistically significant difference in absolute error between students and residents using the 3D interface ($-0.22 \pm 0.7°$, $t(27) = -0.320$, $p = 0.751$, $d = 0.12$) and the hybrid interface ($0.34 \pm 0.51°$, $t(23.901) = 0.657$, $p = 0.517$, $d = 0.2$).

Effects of interface and experience on SUS score

The mean SUS score for both groups of subjects is shown in figure 5.4. For the student group, the SUS score was statistically significantly lower for the 2D interface compared to the 3D interface ($M = 14.08$, $SE = 3.87$, $p = 0.006$) and for the 2D interface compared to the hybrid interface ($M = 13.29$, $SE = 4.45$, $p = 0.024$). The SUS score was not statistically significantly different between the 3D and hybrid interfaces ($M = 0.79$, $SE = 1.47$, $p = 1.0$).

For the resident group, the SUS score was statistically significantly higher for the 2D interface compared to the 3D interface ($M = 11.8$, $SE = 3.25$, $p = 0.016$) and for the 2D interface compared to the hybrid interface ($M = 6.8$, $SE = 1.79$, $p = 0.013$). The SUS score was not statistically significantly different between the 3D and hybrid interfaces ($M = 5.0$, $SE = 2.44$, $p = 0.213$).

The SUS score of students when compared to residents was statistically significantly lower for the 2D interface (8.83 ± 4.23, $t(27) = 2.085$, $p = 0.047$, $d = 0.82$). The SUS score was statistically significantly higher for the 3D interface (17.05 ± 3.2, $t(27) = 5.336$, $p < 0.0005$, $d = 2.08$) and the hybrid interface (11.26 ± 3.56, $t(27) = 3.166$, $p = 0.004$, $d = 1.24$).

Effects of interface and experience on physical task load

The mean physical task load for both groups of subjects is shown in figure 5.5. For the student group, the physical load was statistically significantly lower for the 2D interface compared to 3D ($M = 10.0$, $SE = 1.83$, $p < 0.0005$) and for the

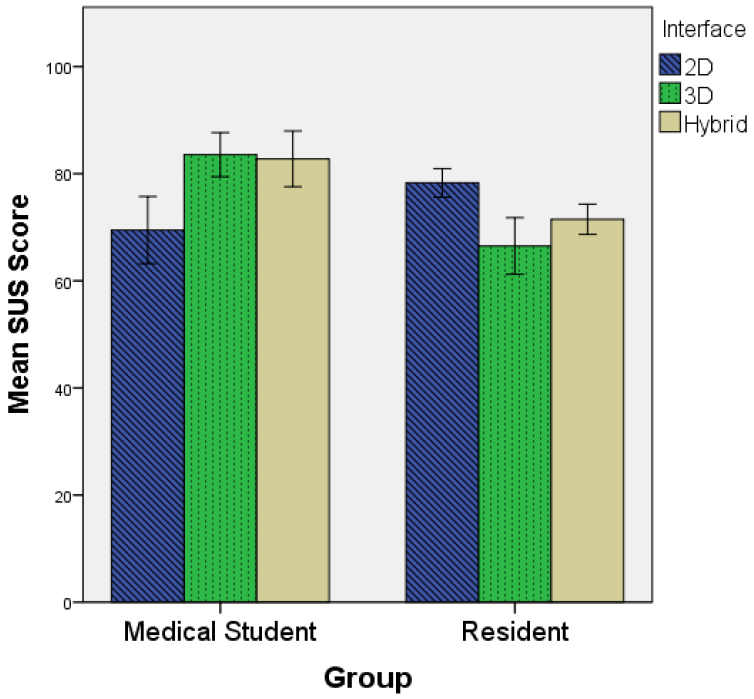


Figure 5.4: Mean SUS score

2D interface compared to the hybrid interface ($M = 6.05$, $SE = 2.25$, $p = 0.045$). The physical task load was not statistically significantly different between the 3D and hybrid interfaces ($M = 3.947$, $SE = 2.08$, $p = 0.221$).

For the residents group, the physical task load was statistically significantly lower for the 2D interface compared to the 3D interface ($M = 21.0$, $SE = 3.06$, $p < 0.0005$) and for the 2D interface compared to the hybrid interface ($M = 8.5$, $SE = 2.59$, $p = 0.028$) but the physical task load was statistically significantly higher for 3D interface compared to the hybrid interface ($M = 12.5$, $SE = 2.5$, $p = 0.002$).

The physical task load for students when compared to residents was statistically significantly lower for the 3D interface (12.18 ± 2.71, $t(27) = 4.504$, $p < 0.0005$, $d = 1.76$). There was no statistically significant difference in physical task load between students and residents for the 2D (1.18 ± 2.13, $t(27) = 0.555$, $p = 0.583$, $d = 0.22$) and hybrid (3.63 ± 2.92, $t(27) = 1.244$, $p = 0.224$, $d = 0.49$) interfaces.

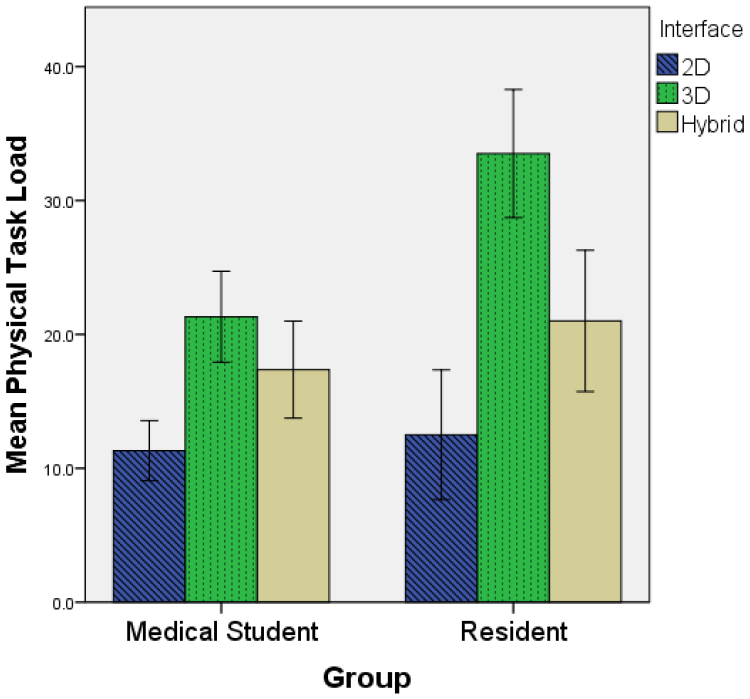


Figure 5.5: Mean physical task load

5.3.2 Main Effects

The main effects of the *interface* and *experience* on the dependent variables *completion time*, *absolute error*, *SUS score*, *physical task load*, and *mental task load* were also investigated. For the interface, since there were three conditions (*2D*, *3D*, and *hybrid*), the Bonferroni correction was used to adjust the p-value for statistical bias.

Effects of the interface on completion time

The main effect of the interface showed a statistically significant difference in mean completion time for different interfaces ($F(2, 54) = 21.716$, $p < 0.0005$, partial $\eta^2 = 0.446$). The mean completion time for the three interface conditions is shown in figure 5.6.

The mean completion time was statistically significantly lower for the 2D interface compared to the 3D interface ($M = 44.55\,seconds$, $SE = 7.99$, $p < 0.0005$), while it was statistically significantly higher for the 3D interface compared to the

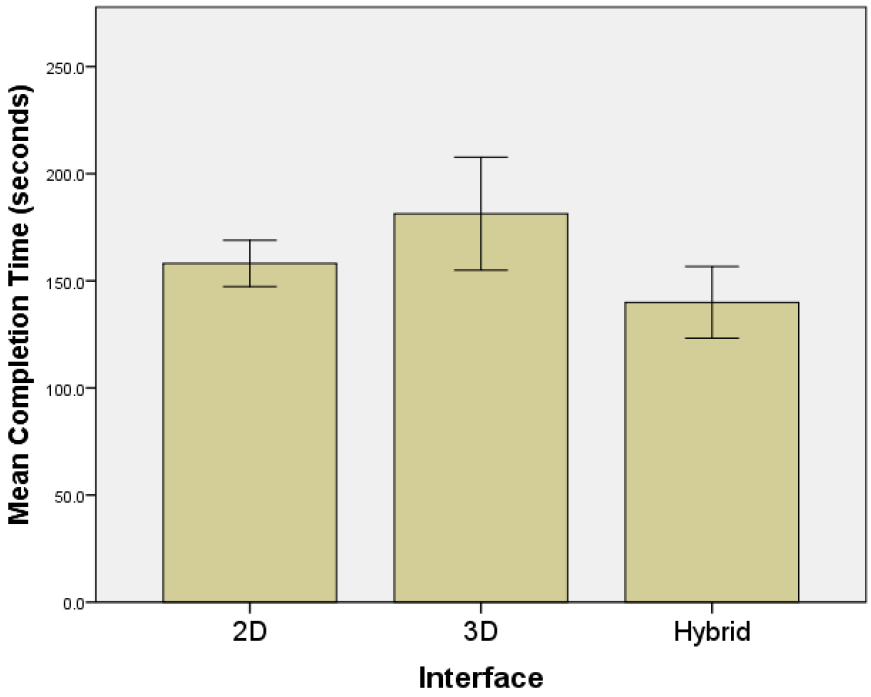


Figure 5.6: Effect of interface on Mean completion time

hybrid interface ($M = 47.83\,seconds$, $SE = 8.88$, $p < 0.0005$). The mean completion time was not statistically significantly different between 2D and hybrid interfaces ($M = 3.28\,seconds$, $SE = 7.38$, $p = 1.0$).

Effects of the interface on absolute error

The main effect of the interface showed a statistically significant difference in mean absolute error for different interfaces ($F(2, 54) = 16.873$, $p < 0.0005$, partial $\eta^2 = 0.385$). The mean absolute error for the three interface conditions is shown in figure 5.7.

The mean absolute error was statistically significantly higher for the 2D interface compared to the 3D interface ($M = 2.60°$, $SE = 0.656$, $p = 0.001$) and 2D interface compared to the hybrid interface ($M = 3.379°$, $SE = 0.63$, $p < 0.0005$), but the mean absolute error was not statistically significantly different between the 3D and hybrid interfaces ($M = 0.782°$, $SE = 0.54$, $p = 0.478$).

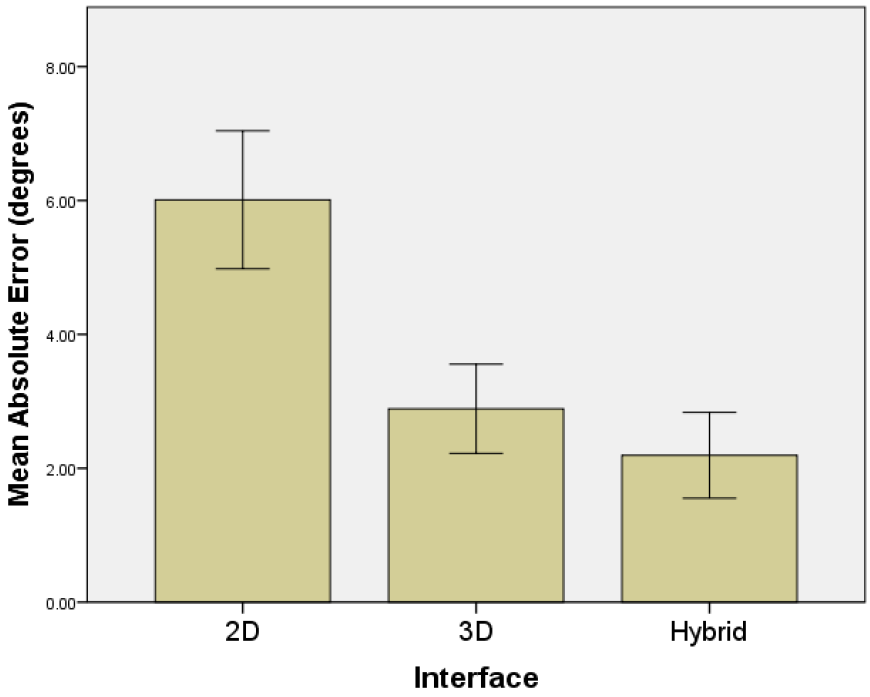


Figure 5.7: Effect of interface on mean absolute error

Effects of the interface on SUS score

The main effect of the interface showed no statistically significant difference in mean SUS score for different interfaces ($F(1.288, 34.772) = 0.799$, $p = 0.407$, partial $\eta^2 = 0.029$, $\epsilon = 0.644$).

Effects of the interface on physical task load

The main effect of the interface showed a statistically significant difference in mean physical task load for different interfaces ($F(2, 54) = 40.125$, $p < 0.0005$, partial $\eta^2 = 0.598$). The mean physical task load for the three interface conditions is shown in figure 5.8.

The mean physical task load was statistically significantly lower using the 2D interface compared to the 3D interface ($M = 15.5$, $SE = 1.68$, $p < 0.0005$) and the 2D interface compared to the hybrid interface ($M = 7.28$, $SE = 1.82$, $p = 0.001$), but the mean physical task load was statistically significantly higher for the 3D interface compared to the hybrid interface ($M = 8.23$, $SE = 1.70$, $p < 0.0005$).

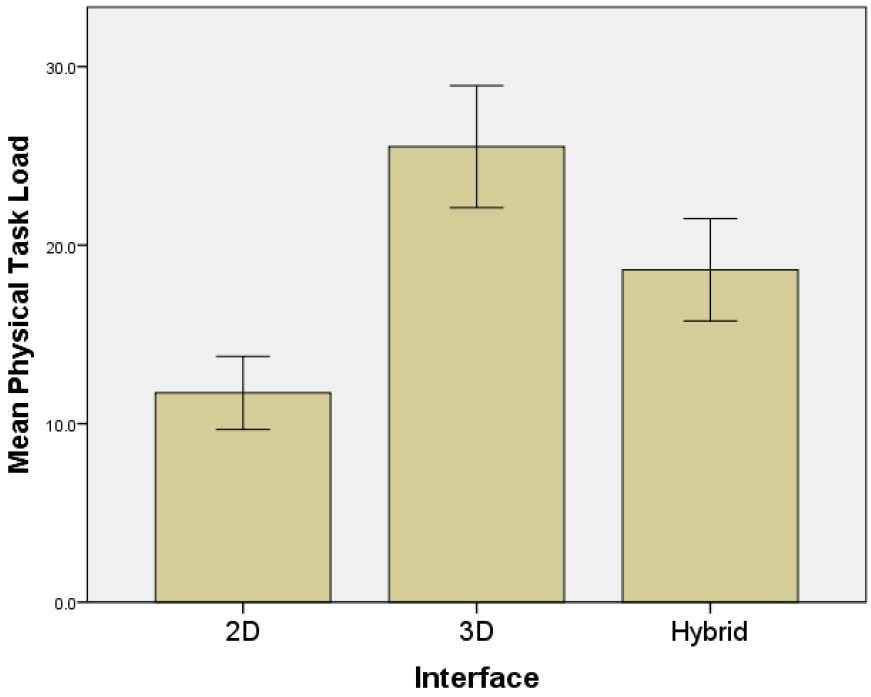


Figure 5.8: Effect of interface on mean physical task load

Effects of the experience on completion time

The main effect of the experience showed that there was a statistically significant difference in mean completion time between experience groups ($F(1, 27) = 30.149$, $p < 0.0005$, partial $\eta^2 = 0.528$). The mean completion time for students was statistically significantly lower than the residents ($M = 51.82\,seconds$, $SE = 9.44$, $p < 0.0005$). The mean completion time for both experience groups is shown in figure 5.9.

Effects of the experience on absolute error

The main effect of the experience showed that there was a statistically significant difference in mean absolute error between experience groups ($F(1, 27) = 15.135$, $p = 0.001$, partial $\eta^2 = 0.359$). The mean absolute error for students was statistically significantly higher than the residents ($M = 1.089°$, $SE = 0.28$, $p = 0.001$). The mean absolute error for both experience groups is shown in figure 5.10.

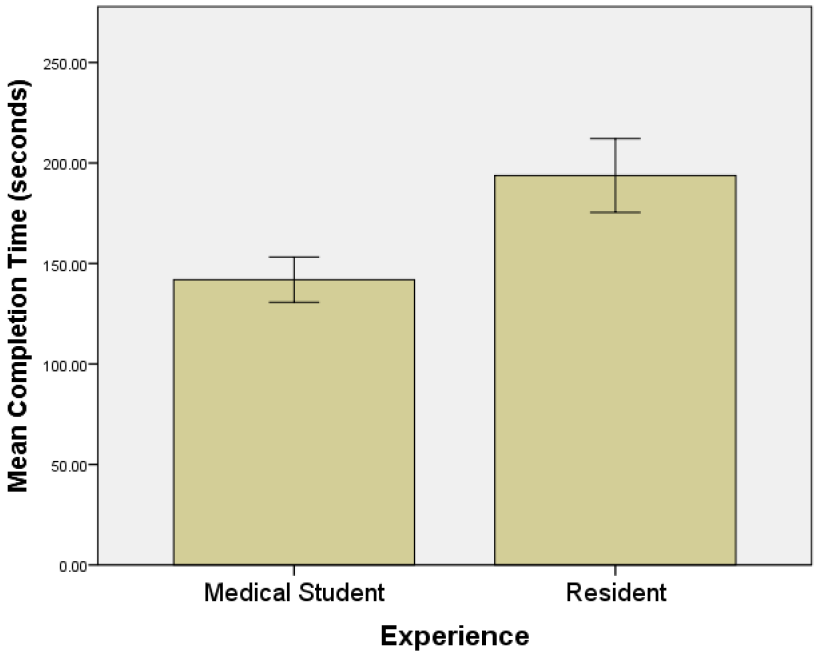


Figure 5.9: Effect of experience on mean completion time

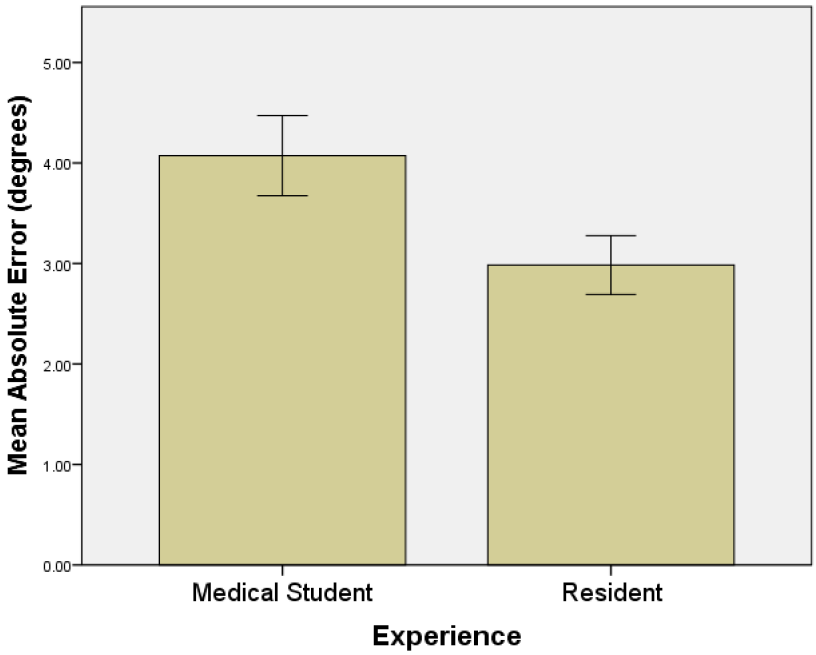


Figure 5.10: Effect of experience on mean absolute error

Effects of the experience on SUS score

The main effect of the experience showed that there was a statistically significant difference in mean SUS score between experience groups ($F(1, 27) = 9.263$, $p = 0.005$, partial $\eta^2 = 0.255$). The mean SUS score for students was statistically significantly higher than the residents ($M = 6.50$, $SE = 2.14$, $p = 0.005$). The mean SUS score for both experience groups is shown in figure 5.11.

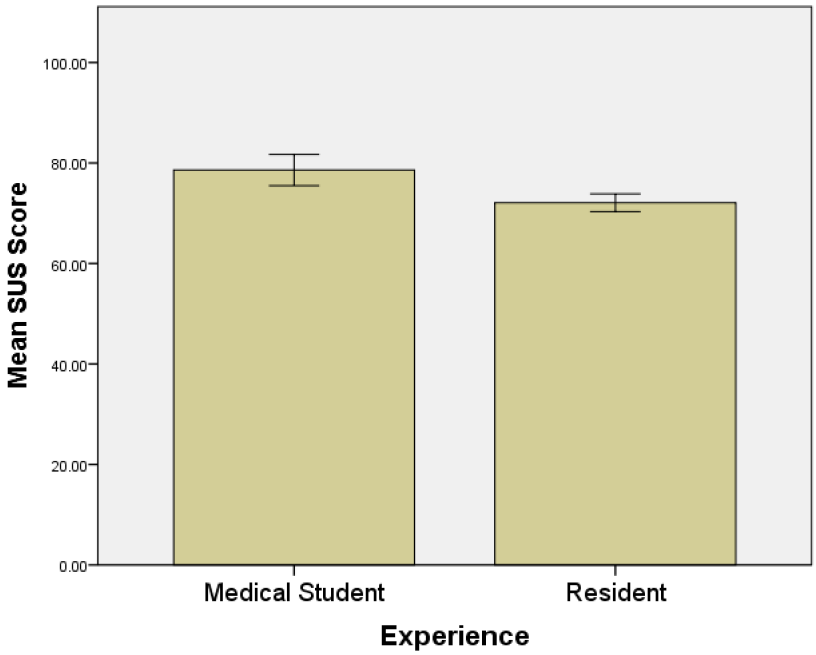


Figure 5.11: Effect of experience on mean SUS score

Effects of the experience on physical task load

The main effect of the experience showed that there was a statistically significant difference in mean physical task load between experience groups ($F(1, 27) = 11.463$, $p = 0.002$, partial $\eta^2 = 0.298$). The mean physical task load for students was statistically significantly lower than the residents ($M = 5.67$, $SE = 1.67$, $p = 0.002$). The mean physical task load for both experience groups is shown in figure 5.12.

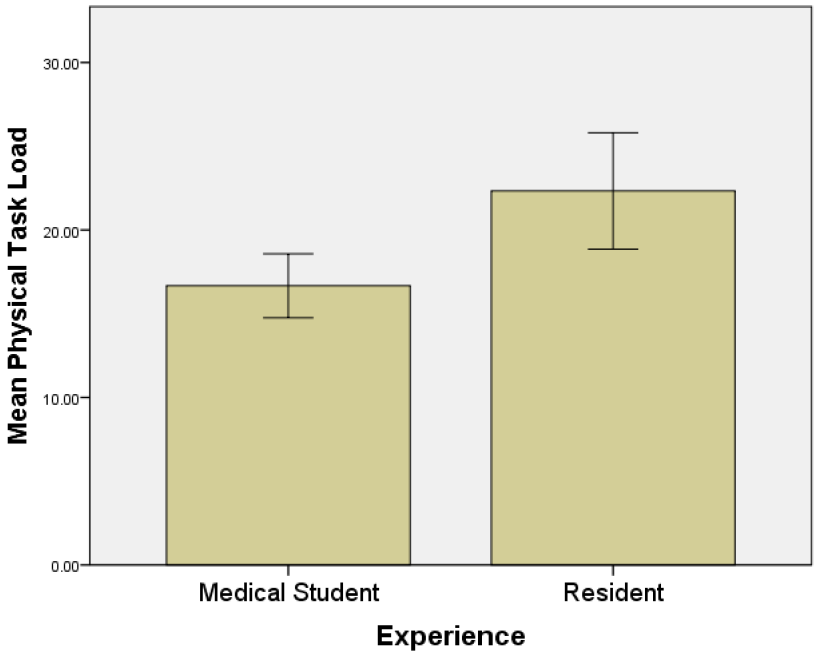


Figure 5.12: Effect of experience on mean physical task load

5.3.3 *Mode changes with the 3D interface*

The average mode changes were compared using the 3D interface between students ($M = 4.58$, $SD = 1.07$) and the residents ($M = 7.5$, $SD = 3.28$) by running the independent samples t-test. The number of mode changes by residents was statistically significantly higher than students (2.92 ± 0.81, $t(27) = 3.590$, $p = 0.001$, $d = 1.40$). The mean mode changes for both experience groups is shown in figure 5.13.

5.3.4 *Age correlations with 3D interface*

A strong positive correlation was found between age and completion time using the 3D interface ($r(29) = 0.715$, $p < 0.0005$). Also, a strong negative correlation was found between age and SUS score for the 3D interface ($r(29) = -0.623$, $p < 0.0005$). However, there was no statistically significant correlation between age and task accuracy ($r(29) = -0.027$, $p = 0.889$).

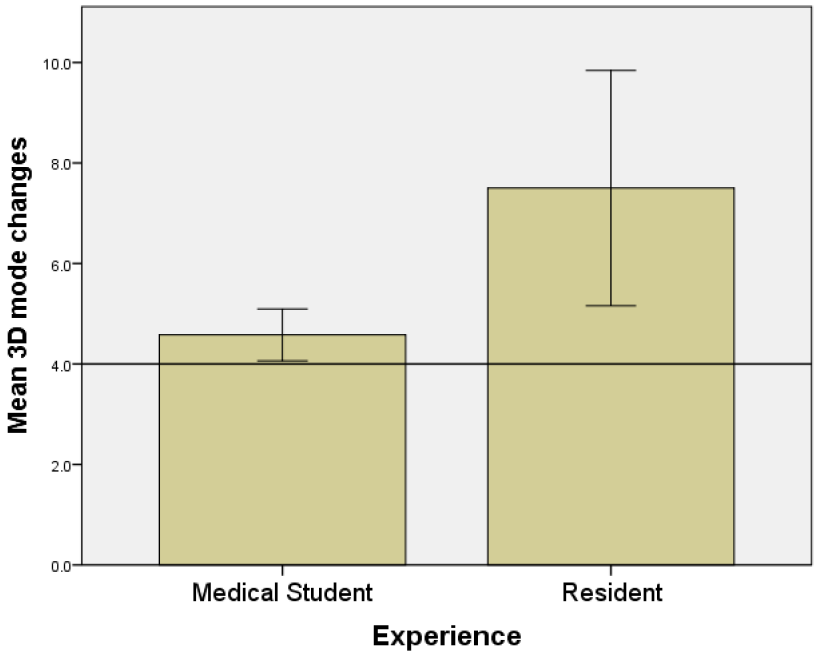


Figure 5.13: Difference in the mean 3D mode changes (reference line at 4.0 shows the minimum mode changes required for the task)

5.3.5 Participant feedback from the interview

All participants appreciated the availability of the 3D model on the zSpace display. They felt that it offered a quick and complete view of the patient's anatomy. Most medical students reported feeling more confident while performing the diagnosis task in 3D and hybrid conditions as the 3D model helped them orient the slice quickly. They also reported that 3D rotation of the arbitrary plane using the 2D mouse was difficult, and guidelines on other orthogonal planes were hard to follow and confusing.

The residents reported feeling more confident in the 2D and hybrid conditions, as they found it easier to use the 2D slice guidelines to verify the integrity of their slice orientation. All of them reported difficulty performing precise annotations and measurements with the stylus in the 3D condition. They suggested that some form of stabilization or scaling was essential for using the stylus.

Most participants felt that the mouse was more accurate for annotation and measurement steps. All participants felt that learning to use the stylus was not only very natural but straightforward and easy. They reported finding it difficult

to perform annotation and measurement using the 3D interface due to unsteady hands while using the stylus. A few participants mentioned that the limited virtual stylus length forced them to reach further and thus made them experience more hand fatigue.

5.3.6 Observations

The stylus seemed very intuitive as no participant had any difficulty manoeuvring it. In the 3D condition, some participants felt that it was difficult to perform precise tasks such as annotation and measurement using the stylus. To improve hand stability, most participants rested their elbow on the table, and some participants tried using their non-dominant hand to support their dominant hand.

All participants seemed to enjoy the stylus input and stereoscopic display. Near the end of their practice session, they often reloaded the dataset and explored the anatomical features of the 3D model using the stylus, in the 3D and hybrid conditions. While in the 2D condition, they would proceed to start the evaluation task. Some participants even requested additional time at the end of their experiment, to further explore the 3D model.

In the 2D condition, medical students seemed confused by the slice guidelines on other orthogonal planes. They found it difficult to make precise 3D rotations using these guidelines. Medical students appeared more comfortable using the 3D interface compared to residents. The resident group seemed to have difficulty remembering the location and functions of 3D buttons in the 3D interface. They referred to the 3D control reference card more, compared to medical students.

5.4 Discussion

Each participant only received five minutes of training followed by five minutes of practice time to familiarize themselves with the interface. Despite this, they were able to complete the task in less than five minutes with any interface. This showed that not much training was required to learn the interface and tools necessary for performing the diagnosis task.

The fastest condition for students was the hybrid interface followed by the 3D and 2D interfaces. They were also more accurate with the hybrid and 3D interfaces compared to 2D. The medical students' slow and inaccurate performance with the

2D interface could probably be attributed to their inexperience with 2D tools, and the difficulty of performing 3D rotation of the slice plane using the 2D mouse input.

Using the hybrid interface, students were able to achieve the same performance as the more experienced users. I believe that the 3D component provided additional context for the task, which mitigated the need for students to mentally visualize and understand the anatomy. Since they had no prior experience with any interface, it shows that they were not only able to learn the hybrid interface quickly but were also to use it very efficiently.

While their accuracy using the 2D interface was low, using the hybrid interface, their accuracy improved significantly and was even comparable to that of the residents. The fact that using the hybrid interface, medical students were able to perform as well as the radiology residents in terms of time and accuracy is a significant finding. This implies that the hybrid interface could reduce the training time needed to get to same skill level. This could have major implications for improving diagnostic radiology training.

The fastest condition for residents was the 2D interface followed by the hybrid and 3D interfaces. However, surprisingly, the residents were most accurate using the hybrid interface followed by the 3D and 2D interfaces. Since the residents had been using the 2D interface daily for over a year, it is no surprise that they were fastest with the 2D interface and significantly outperformed the students with it. Their prior expertise with the 2D interface made them faster with the 2D portion of the hybrid interface, and so were overall faster with the hybrid interface than the 3D interface.

The lower accuracy of residents using the 2D interface despite their prior experience could be attributed to the difficulty in orienting the arbitrary slice using the 2D mouse input. Such difficulty can result in a habit of considering measurements with potentially lower accuracy, as acceptable.

By providing an easier way to rotate the arbitrary slice in the hybrid interface, despite the habit of "rough measurements", the accuracy has still increased. This is a significant finding. However, it should be noted that the cause could also be the novelty of the hybrid interface, leading to more attention from the residents during the diagnosis task. A future study looking at the use of the interface over multiple sessions could help identify the true cause of this effect.

I observed that residents found it difficult to cope with the different task workflow and relied more heavily on the interface reference cards. The results show that while using the 3D interface, on average, residents made significantly more mode changes than the medical students. Observation from the videos revealed that this was due to mistakes, lack of context awareness, or inadvertent mode changes. This indicated that they found it difficult to follow the task workflow in the 3D condition. The residents did not have this problem in the hybrid condition, as it did not involve any 3D mode changes. This implies that the performance of residents can be improved even more with practice and learning.

The 3D stylus had a one to one mapping of hand motion. While this was very natural for exploring the 3D model, precise movements were difficult. Unsteady hands made it even harder to perform precise interactions. This explains why participants experienced difficulties with annotation and measurements in the 3D condition. The stylus interaction also demanded higher physical hand motion compared to the 2D mouse. This explains the higher physical task load score for the 3D and hybrid interfaces.

A strong positive correlation was observed between age and task completion time, with older participants taking longer to complete the task in the 3D condition. This behaviour is similar to that observed by Zudilova et al. [122] in their study on the effect of age on the performance of 3D rotation of medical volumes using an abstract rotation task. It should be noted, however, that the task in their evaluation was quite different from that presented here. A strong negative correlation between age and the 3D SUS score was also found, with older users rating the 3D interface lower. However, their accuracy scores were not affected by age.

The resident group was the older user group, with ages ranging from 28 to 38 years (see section 5.2.1). Hence, it is possible that the observed correlation is a result of their prior extensive experience with Inteleviewer's 2D interface. It is hard to unlearn certain behaviours, that are similar in context but differ in interaction.

Feedback from the subjects in the unstructured interview, showed that most participants preferred the hybrid interface among the three conditions, since it combined the 3D model (showing the patient anatomy), the stylus input for 3D slice manipulation, the synchronized 2D view for verification, and the familiar and precise 2D tools (for annotations and measurements).

The medical students, in particular, appreciated having the 3D model, since it helped them better understand the patient's anatomy, compared to the 2D interface. This is likely the reason why they gave the lowest SUS score for the 2D interface and a relatively high SUS score for the 3D and hybrid interfaces despite the stylus precision issues. The residents gave the lowest SUS score to the 3D interface due to the change in task workflow, and stylus precision problems for the annotations and measurements. Their SUS score for the 2D interface was higher, but this was expected due to their familiarity with the system.

A baseline SUS score of 68 is often used to determine major usability issues with a user interface [117]. The SUS scores for all conditions of the study were higher than this baseline score, within the margin of error. This shows that there were no major usability issues with the interface functionality tested.

The stereoscopic 3D is a key component of the 3D interface. It was required to simulate the holographic 3D experience by rendering the 3D model in the negative parallax region. The scene contrast, eye separation, and scene depth were within the optimal range [123]. Since the maximum task completion time did not exceed 5 minutes, it is less likely that users experienced any ill effects from stereoscopic 3D. Also, there was no such feedback from the users, that would suggest otherwise.

One of the issues mentioned by the users was the mapping of buttons on the 3D stylus. While some participants were satisfied with the way stylus buttons were mapped, others preferred that the left and right buttons on the stylus to be mapped similar to a mouse, even though this was not ergonomic. The left button on the stylus is the hardest to reach with the index finger. Despite this, participants wanted the most common functions to be mapped to the left stylus button, as this would be similar to the traditional mouse (where most functions are performed with the left mouse button).

5.5 Limitations

There were a limited number of radiology residents at the Christchurch Hospital, hence the sample size for the resident group was relatively small. The scope of the diagnosis task used for evaluation was limited. A diagnosis task that required some level of 3D manipulation was chosen to fairly evaluate the system. Only a limited number of diagnostic radiology tasks require 3D manipulation. Further

studies can be run with different diagnostic radiology tasks, possibly with a larger pool of residents to observe differences in performance. Additional experimental data could be captured such as hand and head tracking to study user behaviour in more detail.

As the user study was advertised openly to fourth year medical students using flyers and emails, it is possible that only high achieving students, or those with a particular interest in radiology, might sign up for the user study leading to self-selection bias. The results of usability obtained from SUS may be influenced by the novelty factor of the hybrid interface. Another factor that could influence SUS scores might be the social desirability bias. This is especially true for this evaluation, where the person conducting the user study is also the developer of the interface.

Although the task followed an instructional design which is known to reduce cognitive load for novice users [124], it is possible that the cognitive load for the task could have had a negative impact for some participants. Since the results for mental task load were not statistically significant, it is difficult to know the effect cognitive load on the study.

5.6 Summary

This chapter discussed the evaluation the hybrid system. The key goal was to test the effectiveness of the hybrid interface for diagnostic radiology compared to a standard 2D system. The radiology diagnosis workflow was explored and a methodology to evaluate the system was designed. Later, the choice of participants, the evaluation task, and a detailed explanation of the experiment setup was discussed. The key performance and usability measures were discussed along with a data analysis strategy.

Finally, a statistical analysis of the evaluation results was presented, followed by a detailed discussion of these results as well as some limitations of the study. In the next chapter, the impact of this thesis is discussed in detail.

Chapter VI

Discussion and conclusion

In this chapter, I discuss the impact from the study and other work described in this thesis in the context of the overall thesis goal of improving 3D manipulation for spectral CT data exploration. Finally, I state some recommendations for future work and conclude the findings.

6.1 Discussion

There were many features that were developed during the course of this thesis. Some of the key features include:

- The development of a two-axis valuator method for mouse-based 3D manipulation along with several modifications to the algorithm to accommodate the use of various volume visualisation tools along with the method.

- The development of a GPU-based rapid mesh extraction method based on the Marching Cubes algorithm, including the development of a novel method for the efficient computation of gradient normals.

- The support for stereoscopic 3D rendering for both the volumetric images from the MARS MCRT algorithm as well as the extracted mesh.

- The development of a real-time rendering engine capable of displaying volumetric images from the MARS MCRT algorithm or a 3D mesh extracted from the volume along with other 3D models in a 3D scene.

- The development of the arbitrary slice view to support an oblique slicing plane for data exploration.

There were two key benefits from these features. Firstly, they improved various interactivity aspects of MARS Vision addressing the limitations described in Chapter 2. Secondly, these features provided a framework to facilitate future research and were used to implement the SpaceMouse input, the zSpace interface, and eventually the hybrid user interface described in Chapter 4. They were also essential to conduct the evaluation study described in Chapter 5.

The improvements to mouse-based 3D manipulation helped the pre-clinical teams to visualize MARS spectral CT data for their research and produce images for their publications. These include journal articles showing the imaging of a lamb tissue dataset (Aamir et al. [125]) and demonstrating the reduction of beam hardening using narrow energy-bins in spectral CT (Rajendran et al. [40]), for which I was also a co-author. Other publications include visualizing a mouse with gold nano-particles to show spectral CT data acquisition (Walsh et al. [126]), and other visualizations to demonstrate both the quantitative imaging of osteoarthritic cartilage (Rajendran et al. [41]) and other clinical applications of spectral CT (Anderson and Butler [4]).

During the course of my Ph.D. I was also a consultant for knowledge transfer, participating in discussion of various physical models to improve MARS scanner calibration. This would improve the quality of spectral CT datasets, consequentially improving visualization and diagnosis. This resulted in two published journal articles including the per-pixel calibration of photon counting detectors (Atharifard et al. [127]) and energy response calibration using oblique fluorescence in MARS scanner (Broeke et al. [128]), for which I was a co-author.

The evaluation study provided empirical evidence to show that the hybrid interface was more efficient for novice users, and more accurate for both novice and experienced users when compared to the 2D interface. This is a significant finding because it indicates that as the interaction techniques mature, hybrid interfaces can provide significant benefit to CT exploration and diagnostic tasks. I would expect similar benefits for MARS users who are currently using the hybrid interface for exploration of spectral CT datasets.

The interface was very easy to learn evidenced by the fact that all users were able to perform the scoliosis diagnosis after a five-minute training session. A significant finding from the evaluation study was that novice users were not only able to learn the hybrid interface quickly but also to use it very efficiently to perform a

diagnosis task with an accuracy that was comparable to that of more experienced users. This implies that the hybrid interface could be a better alternative to the 2D only interface for diagnostic radiology training.

Some MARS users reported that they were able to locate and measure a tumour in spectral CT scan of a mouse using the hybrid interface in less than 5 minutes. Before, when using the 2D only tools, the time taken was about 30 minutes instead. This is mainly due to the better 3D manipulation provided by the hybrid interface. They were able to position the arbitrary slice much faster and were able to later perform 2D measurements on the slice. (see figure 6.1).

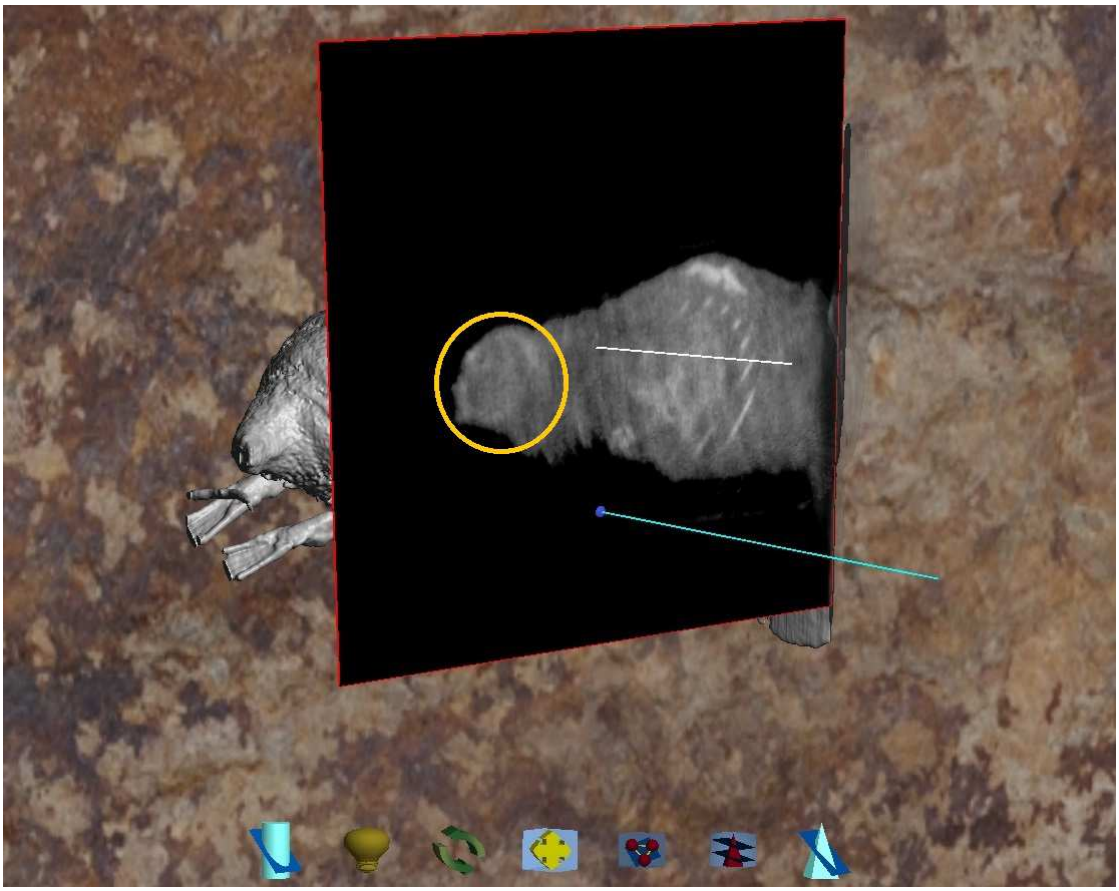


Figure 6.1: An image showing the use of the 3D component of the hybrid interface for locating the tumour on an arbitrary slice in the MARS scan of a mouse. The tumour has been enhanced with a yellow circle.

By providing efficient ways for 3D exploration while still preserving the 2D interaction techniques for more precise tasks, the hybrid interface is a good al-

ternative to the standard 2D only interfaces. Since the hybrid interface has been integrated into MARS Vision (see figure 6.2), it is currently available to all MARS users for their research. The hybrid interface has since also become an integral part of the MARS product.

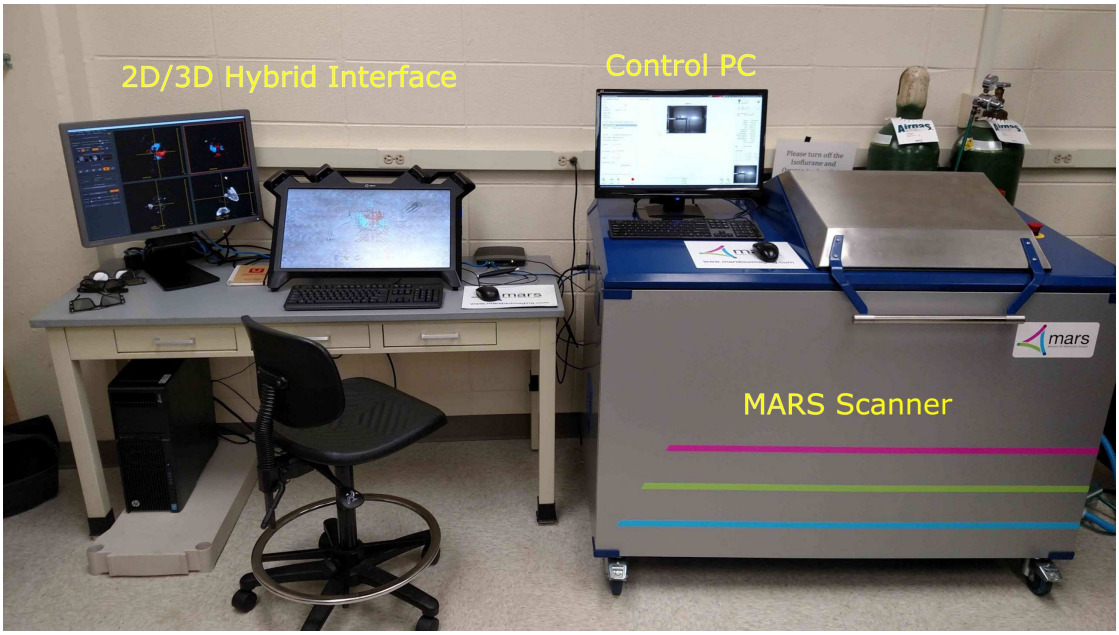


Figure 6.2: An image showing the current MARS small bore scanner product along with my hybrid interface running MARS Vision to the left.

The research presented in this thesis has strong implications on the future direction for medical visualization and diagnosis software. The 3D input is shown to provide better 3D manipulation that could significantly improve the accuracy for some diagnosis tasks. However, most users have years of training with the 2D mouse input and it is hard to welcome change.

By providing a hybrid interface, it is possible to still support the traditional 2D workflow without the need for an immediate change. This allows the users to take advantage of 3D input optionally as and when required. It also opens the possibility for users to start finding ways of combining the 2D and 3D components to perform the diagnostic tasks more efficiently. The hybrid approach could be the way for future diagnostic software.

6.2 Future Work

Since the evaluation was based on direct manipulation and wand style interaction for the stylus, a future study could focus on the efficiency of the tethered rotation. The study from Saalfeld et al. [93] evaluated a modified version of my tethered rotation with computer scientists. I believe that it would be beneficial to conduct a study replicating a more realistic scenario with more experienced participants such as physicians or radiologists.

Another direction could be to evaluate the laser and wand styles for a variety of tasks to determine their strengths and weaknesses. Perhaps a hybrid style could prove to be more efficient than either of them. It would also be good to have more longitudinal studies to see how people's performance changes over time, and what learning effects there are with the technology.

The users reported in the study that the zSpace stylus was a very intuitive tool. So it can be expected that future tools designed using the stylus will largely benefit from the intuitive nature of the stylus. One possible direction could involve the implementation of more complex 3D annotation and volume editing tools.

The 3D model in the hybrid interface was intended to give an overview of the anatomy for inexperienced users so that they could quickly and easily understand the position and orientation of a slice, relative to the entire scan. Although a mesh was used for visualizing the anatomy, other volume rendering techniques could be used to better represent soft tissues. This would enable performing other diagnostic tasks that strictly rely on soft tissue exploration using the hybrid interface.

In the 3D condition, lack of stylus stabilization and the fixed virtual stylus length were the issues that negatively impacted task performance and increased fatigue. The 3D stylus precision can be improved by filtering the stylus input or using some form of scaled manipulation such as PRISM [129]. Speech input could be explored to improve the mode switching leading to future studies of speech versus non-speech input. Interaction with the 3D components could be improved by exploring other forms of 3D input such as freehand input.

It would be interesting to explore effects (especially fatigue) while using the interface for a longer period of time as typical radiology sessions last for 30 to 60 minutes. Users would be expected to perform even better with additional training on the hybrid interface, as they already performed relatively well with just five

minutes of training in the evaluation study.

Since the hybrid interface is more beneficial to novice users, it can have a major implication in training. This motivates the use of the hybrid interface as a training tool for diagnostic radiology. A user study could be designed and conducted for trainees at different points of time to assess the efficiency of the hybrid interface as a training tool.

The novice users stated that they felt more confident using the hybrid interface since they could refer to the anatomy of the patient by manipulating the 3D model while performing the diagnosis task. Thus, the interface can be used in a teaching scenario to demonstrate various medical aspects using human CT datasets. The hybrid system is only designed for a single user, so if the teacher is the primary user, it is necessary to share the experience of the teacher with multiple users.

Sharing the experience of the teacher could be done in three ways: fixed scene mirroring, first person scene and third person scene. Fixed scene mirroring method is simply rendering the 3D scene from a fixed camera onto a secondary display such as a projector. This can be done by rendering the scene in stereoscopic 3D, or simply 2D for a standard 2D display. First person scene method renders the scene from the presenter's perspective, taking their head movements into consideration.

The third person scene method renders the scene from a camera position behind the user. The zSpace company also offers an augmented reality setup called zView. This setup uses a web-cam located behind the user and shows the 3D scene from a third person (web-cam) view overlaid on top of the physical 3D display (see figure 6.3).

This is an improvement to the third person scene method which overlays the scene in the real world. This method is currently being used for teaching in some primary schools. A combination of the zView and the hybrid interface could be used for teaching and demonstration. It can also be used to make videos tutorials showing the workflow for exploring the MARS spectral CT datasets.

The work from this thesis improves the efficiency of 3D exploration by designing the hybrid user interface. This also relates strongly to the field of pre-surgical planning. For example, the orthopaedic surgeons often use 3D printing to examine broken bone segments in order to plan their surgery. There is a clear opportunity to use the 3D component of the hybrid interface in place of physical 3D printed bone segments for such tasks along with a reliable surgical simulation tool. This

Figure 6.3: An example image showing the functionality of zView [86]. Used with permission.

could have significant benefits to surgical planning.

6.3 Conclusion

To summarize, this research has:

- Explored the interactivity limitations of MARS Vision as at the start of my Ph.D. with an emphasis on 3D manipulation and formulated a set of requirements to address the limitations and facilitate further research.

- Implemented an effective 3D manipulation method based on the comparison in literature. Modified the algorithm to solve some issues caused by the type of 3D manipulation and other tools in MARS Vision.

- Implemented various features to facilitate further research including the stereoscopic 3D rendering, the rapid mesh extraction, the arbitrary slice and a real-time rendering engine.

- Explored literature for 3D input devices and implemented the SpaceMouse and the zSpace interface. Designed a novel 2D/3D hybrid user interface combining the traditional 2D system and the zSpace interface.

- Evaluated the hybrid interface against the 2D interface in a diagnostic radiology scenario for experienced and novice users. The evaluation proved the effectiveness of the hybrid interface over the 2D interface.

- Integrated the interface into MARS Vision to make it available to all MARS users. The interface can be used in conjunction with all the other visualisation tools to explore MARS spectral datasets.

In conclusion, the work presented in this thesis advanced the field of medical visualisation by the development of an efficient hybrid interface for medical image evaluation/diagnosis and proving a platform for future research. This could also have a significant impact in the field of medical diagnosis with implications for spectral CT with the MARS project. The interface is also integrated into the MARS product enabling its many pre-clinical researchers to take advantage of more efficient 3D exploration.

The developments in the interactive exploration of datasets will allow the users to access more of the full potential of the MARS spectral system, consequentially resulting in more efficient and accurate results. As the MARS project proceeds towards its goal of developing a spectral CT scanner for human imaging, it will have a significant impact on health care and medicine.

References

[1] C. P. Botha, B. Preim, A. Kaufman, S. Takahashi, and A. Ynnerman, "From individual to population: Challenges in Medical Visualization," *arXiv preprint arXiv:1206.1148*, 2012.

[2] J. M. Hoffman and A. E. Menkens, "Molecular imaging in cancer: future directions and goals of the national cancer institute," *Academic Radiology*, vol. 7, no. 10, pp. 905–907, 2000.

[3] J. Fornaro, S. Leschka, D. Hibbeln, A. Butler, N. Anderson, G. Pache, H. Scheffel, S. Wildermuth, H. Alkadhi, and P. Stolzmann, "Dual-and multi-energy CT: approach to functional imaging," *Insights into imaging*, vol. 2, no. 2, pp. 149–159, 2011.

[4] N. G. Anderson and A. P. Butler, "Clinical applications of spectral molecular imaging: potential and challenges," *Contrast Media & Molecular Imaging*, vol. 9, no. 1, pp. 3–12, 2014.

[5] K. Taguchi and J. S. Iwanczyk, "Vision 20/20: Single photon counting x-ray detectors in medical imaging," *Medical physics*, vol. 40, no. 10, 2013.

[6] "MARS Bioimaging Ltd.." http://www.marsbioimaging.com/mars/. Accessed: 2017-01-09.

[7] R. Zainon, J. Ronaldson, T. Janmale, N. Scott, T. Buckenham, A. Butler, P. Butler, R. Doesburg, S. Gieseg, and J. Roake, "Spectral CT of carotid atherosclerotic plaque: comparison with histology," *European radiology*, vol. 22, no. 12, pp. 2581–2588, 2012.

[8] J. P. Ronaldson, *Quantitative soft-tissue imaging by spectral CT with Medipix3*. Thesis, University of Otago, 2012.

[9] K. B. Berg, J. M. Carr, M. J. Clark, N. J. Cook, N. G. Anderson, N. J. Scott, A. P. Butler, P. H. Butler, A. P. Butler, S. C. Hendy, *et al.*, "Pilot study to confirm that fat and liver can be distinguished by spectroscopic tissue response on a medipix-all-resolution system-ct (mars-ct)," in *AIP Conference Proceedings*, vol. 1151, pp. 106–109, AIP, 2009.

[10] N. Anderson, A. Butler, N. Scott, N. Cook, J. Butzer, N. Schleich, M. Firsching, R. Grasset, N. de Ruiter, and M. Campbell, "Spectroscopic (multi-energy) CT distinguishes iodine and barium contrast material in MICE," *European radiology*, vol. 20, no. 9, pp. 2126–2134, 2010.

[11] M. Firsching, A. P. Butler, N. Scott, N. G. Anderson, T. Michel, and G. Anton, "Contrast agent recognition in small animal ct using the medipix2 detector," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 607, no. 1, pp. 179–182, 2009.

[12] R. P. McMahan, D. Gorton, J. Gresock, W. McConnell, and D. A. Bowman, "Separating the effects of level of immersion and 3D interaction techniques," in *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 108–111, ACM, 2006.

[13] K. Hinckley, J. Tullio, R. Pausch, D. Proffitt, and N. Kassell, "Usability analysis of 3D rotation techniques," in *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pp. 1–10, ACM, 1997.

[14] M. Chen, S. J. Mountford, and A. Sellen, "A study in interactive 3-D rotation using 2-D control devices," *ACM SIGGRAPH Computer Graphics*, vol. 22, no. 4, pp. 121–129, 1988.

[15] A. I. Chernoglasov, *Tools for Visualising MARS Spectral CT Datasets*. PhD thesis, University of Canterbury, 12 2016.

[16] R. E. Cooke Jr, M. G. Gaeta, D. M. Kaufman, and J. G. Henrici, "Picture archiving and communication system," June 3 2003. US Patent 6,574,629.

[17] S. Nik, J. Meyer, and R. Watts, "Optimal material discrimination using spectral x-ray imaging," *Physics in medicine and biology*, vol. 56, no. 18, p. 5969, 2011.

[18] J. Schindelin, C. T. Rueden, M. C. Hiner, and K. W. Eliceiri, "The imagej ecosystem: An open platform for biomedical image analysis," *Molecular reproduction and development*, vol. 82, no. 7-8, pp. 518–529, 2015.

[19] R. A. Novelline and L. F. Squire, *Squire's fundamentals of radiology*. La Editorial, UPR, 2004.

[20] M. A. Hurrell, A. P. H. Butler, N. J. Cook, P. H. Butler, J. P. Ronaldson, and R. Zainon, "Spectral hounsfield units: a new radiological concept," *European radiology*, vol. 22, no. 5, pp. 1008–1013, 2012.

[21] P. Mildenberger, M. Eichelberg, and E. Martin, "Introduction to the dicom standard," *European radiology*, vol. 12, no. 4, pp. 920–927, 2002.

[22] "Dicom part 10: Media storage and file format for media interchange," 2014.

[23] Y. Liu, K. D. Hopper, D. T. Mauger, and K. A. Addis, "Ct angiographic measurement of the carotid artery: optimizing visualization by manipulating window and level settings and contrast material attenuation," *Radiology*, vol. 217, no. 2, pp. 494–500, 2000.

[24] T. Kroes, F. H. Post, and C. P. Botha, "Exposure render: An interactive photo-realistic volume rendering framework," *PloS one*, vol. 7, no. 7, p. e38586, 2012.

[25] V. Spitzer, M. J. Ackerman, A. L. Scherzinger, and D. Whitlock, "The visible human male: a technical report," *Journal of the American Medical Informatics Association*, vol. 3, no. 2, pp. 118–130, 1996.

[26] A. Buades, B. Coll, and J.-M. Morel, "The staircasing effect in neighborhood filters and its solution," *IEEE transactions on Image Processing*, vol. 15, no. 6, pp. 1499–1505, 2006.

[27] J. T. Kajiya and B. P. Von Herzen, "Ray tracing volume densities," in *ACM Siggraph Computer Graphics*, vol. 18, pp. 165–174, ACM, 1984.

[28] NVIDIA Corporation, "CUDA Programming guide." http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf, 2010. Accessed: 2013-09-05.

[29] B. Preim, C. Tietjen, W. Spindler, and H. O. Peitgen, "Integration of measurement tools in medical 3D visualizations," in *Proceedings of the conference on Visualization'02*, pp. 21–28, IEEE Computer Society, 2002.

[30] J. G. Hagedorn, J. P. Dunkers, S. G. Satterfield, A. P. Peskin, J. T. Kelso, and J. E. Terrill, "Measurement tools for the immersive visualization environment: Steps toward the virtual laboratory," *Journal of research of the National Institute of Standards and Technology*, vol. 112, no. 5, p. 257, 2007.

[31] R. A. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Information processing letters*, vol. 2, no. 1, pp. 18–21, 1973.

[32] C. R. Salama, "Gpu-based monte-carlo volume raycasting," in *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pp. 411–414, IEEE, 2007.

[33] A. H. Watt and M. Watt, *Advanced animation and rendering techniques*. ACM press New York, NY, USA:, 1992.

[34] L. M. Parsons, "Inability to reason about an object's orientation using an axis and angle of rotation," *Journal of Experimental Psychology-Human Perception and Performance*, vol. 21, no. 6, pp. 1259–1276, 1995.

[35] "Qt API, Th Qt Company." https://www.qt.io/ui/. Accessed: 2017-01-12.

[36] NVIDIA Corporation, "CUDA API." https://developer.nvidia.com/cuda-toolkit. Accessed: 2017-01-12.

[37] B. L. Ken Martin, Will Schroeder, "VTK, The Visualization Toolkit." http://www.marsbioimaging.com/mars/. Accessed: 2017-01-09.

[38] Microsoft, "Direct3D." https://msdn.microsoft.com/en-us/library/windows/desktop/hh769064(v=vs.85).aspx. Accessed: 2017-01-12.

[39] "OpenGL API, Khronos Group." https://www.opengl.org/. Accessed: 2017-01-12.

[40] K. Rajendran, M. Walsh, N. De Ruiter, A. Chernoglazov, R. Panta, A. Butler, P. Butler, S. Bell, N. Anderson, T. Woodfield, *et al.*, "Reducing beam hardening effects and metal artefacts in spectral CT using Medipix3RX," *Journal of Instrumentation*, vol. 9, no. 03, p. P03015, 2014.

[41] K. Rajendran, C. Löbker, B. S. Schon, C. J. Bateman, R. A. Younis, N. J. de Ruiter, A. I. Chernoglazov, M. Ramyar, G. J. Hooper, A. P. Butler, *et al.*, "Quantitative imaging of excised osteoarthritic cartilage using spectral CT," *European radiology*, vol. 27, no. 1, pp. 384–392, 2017.

[42] G. Bell, "Bells trackball," *Source code, formerly at http://www. dispersoid. net/code/trackball. c, available through the Internet Archives copy of July*, vol. 17, p. 2007, 1988.

[43] K. Shoemake, "Arcball: a user interface for specifying three-dimensional orientation using a mouse," in *Graphics Interface*, vol. 92, pp. 151–156, 1992.

[44] Autodesk, "3DS MAX Autodesk Inc.." https://www.autodesk.com/products/3ds-max/overview, 2016. Accessed: 2016-08-02.

[45] R. Bade, F. Ritter, and B. Preim, "Usability comparison of mouse-based interaction techniques for predictable 3D rotation," in *International Symposium on Smart Graphics*, pp. 138–150, Springer, 2005.

[46] K. Henriksen, J. Sporring, and K. Hornbæk, "Virtual trackballs revisited," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 2, pp. 206–216, 2004.

[47] Y. J. Zhao, D. Shuralyov, and W. Stuerzlinger, "Comparison of multiple 3d rotation methods," in *Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2011 IEEE International Conference on*, pp. 1–5, IEEE, 2011.

[48] S. Rybicki, B. DeRenzi, and J. E. Gain, "Usability and performance of mouse-based rotation controllers.," in *Graphics Interface*, pp. 93–100, 2016.

[49] J. B. Kuipers *et al.*, *Quaternions and rotation sequences*, vol. 66. Princeton university press Princeton, 1999.

[50] A. J. Hanson, "Visualizing quaternions," in *ACM SIGGRAPH 2005 Courses*, p. 1, ACM, 2005.

[51] Intelerad, "Inteleviewer Intelerad Medical Systems Incorporated." http://www.intelerad.com/en/products/inteleviewer/, 2017. Accessed: 2017-01-09.

[52] M. Van Beurden, W. IJsselsteijn, and J. Juola, "Effectiveness of stereoscopic displays in medicine: a review," *3D Research*, vol. 3, no. 1, p. 3, 2012.

[53] J. P. McIntire, P. R. Havig, and E. E. Geiselman, "What is 3D good for? a review of human performance on stereoscopic 3D displays," in *SPIE defense, security, and sensing*, pp. 83830X–83830X, International Society for Optics and Photonics, 2012.

[54] J. P. McIntire, S. T. Wright, L. K. Harrington, P. R. Havig, S. N. Watamaniuk, and E. L. Heft, "Optometric measurements predict performance but not comfort on a virtual object placement task with a stereoscopic three-dimensional display," *Optical Engineering*, vol. 53, no. 6, pp. 061711–061711, 2014.

[55] D. A. Southard, "Transformations for stereoscopic visual simulation," *Computers & Graphics*, vol. 16, no. 4, pp. 401–410, 1992.

[56] D. Maupu, M. H. Van Horn, S. Weeks, and E. Bullitt, "3d stereo interactive medical visualization," *IEEE Computer Graphics and Applications*, vol. 25, no. 5, pp. 67–71, 2005.

[57] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *ACM Siggraph Computer Graphics*, vol. 21, pp. 163–169, ACM, 1987.

[58] B. Wuensche, "A survey and analysis of common polygonization methods and optimization techniques," tech. rep., Department of Computer Science, The University of Auckland, New Zealand, 1997.

[59] C. D. Hansen and P. Hinker, "Massively parallel isosurface extraction," in *Proceedings of the 3rd conference on Visualization'92*, pp. 77–83, IEEE Computer Society Press, 1992.

[60] H. Yagou, Y. Ohtake, and A. Belyaev, "Mesh smoothing via mean and median filtering applied to face normals," in *Geometric Modeling and Processing, 2002. Proceedings*, pp. 124–131, IEEE, 2002.

[61] D. Shreiner, B. T. K. O. A. W. Group, *et al.*, *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*. Pearson Education, 2009.

[62] M. Scheuring, "Computer graphics group, university of erlangen, germany." http://lgdv.cs.fau.de/External/vollib/Carp.pvm/, 2013. Accessed: 2013-11-15.

[63] H. Hoppe, "Optimization of mesh locality for transparent vertex caching," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 269–276, ACM Press/Addison-Wesley Publishing Co., 1999.

[64] A. Bogomjakov and C. Gotsman, "Universal rendering sequences for transparent vertex caching of progressive meshes," in *Computer Graphics Forum*, vol. 21, pp. 137–149, Wiley Online Library, 2002.

[65] G. Lin and T.-Y. Yu, "An improved vertex caching scheme for 3d mesh rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 640–648, 2006.

[66] T. Forsyth, "Linear-speed vertex cache optimisation," 2006.

[67] D. H. Eberly, *3D game engine design: a practical approach to real-time computer graphics*. CRC Press, 2006.

[68] T. Whitted, "An improved illumination model for shaded display," in *ACM Siggraph 2005 Courses*, p. 4, ACM, 2005.

[69] D. Bowman, E. Kruijff, J. J. LaViola Jr, and I. P. Poupyrev, *3D User Interfaces: Theory and Practice, CourseSmart eTextbook*. Addison-Wesley, 2004.

[70] Y. Dai, J. Zheng, Y. Yang, D. Kuai, and X. Yang, "Volume-Rendering-Based Interactive 3D Measurement for Quantitative Analysis of 3D Medical Images," *Computational and mathematical methods in medicine*, vol. 2013, 2013.

[71] N. Wilt, *The cuda handbook: A comprehensive guide to gpu programming*. Pearson Education, 2013.

[72] R. Balakrishnan, T. Baudel, G. Kurtenbach, and G. Fitzmaurice, "The Rockin'Mouse: integral 3D manipulation on a plane," in *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pp. 311–318, ACM, 1997.

[73] S. Scali, M. Wright, and A. M. Shillito, "3d modelling is not for wimps," in *Proceedings of HCI international*, 2003.

[74] U. Schultheis, J. Jerald, F. Toledo, A. Yoganandan, and P. Mlyniec, "Comparison of a two-handed interface to a wand interface and a mouse interface for fundamental 3d tasks," in *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pp. 117–124, IEEE, 2012.

[75] K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell, "Passive real-world interface props for neurosurgical visualization," in *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, pp. 452–458, ACM, 1994.

[76] L. D. Cutler, B. Fröhlich, and P. Hanrahan, "Two-handed direct manipulation on the responsive workbench," in *Proceedings of the 1997 symposium on Interactive 3D graphics*, pp. 107–114, ACM, 1997.

[77] B. Frohlich, J. Plate, J. Wind, G. Wesche, and M. Gobel, "Cubic-mouse-based interaction in virtual environments," *Computer Graphics and Applications, IEEE*, vol. 20, no. 4, pp. 12–15, 2000.

[78] L. Gallo, G. De Pietro, and I. Marra, "3D interaction with volumetric medical data: experiencing the wiimote," in *Proceedings of the 1st international conference on Ambient media and systems*, p. 14, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[79] S. Mauser and O. Burgert, "Touch-free, gesture-based control of medical devices and software based on the leap motion controller," *Stud Health Technol Inform*, vol. 196, pp. 265–270, 2014.

[80] J. C. Coelho and F. J. Verbeek, "Pointing task evaluation of leap motion controller in 3d virtual environment," *Creating the Difference*, vol. 78, pp. 78–85, 2014.

[81] P. Apostolellis, B. Bortz, M. Peng, N. Polys, and A. Hoegh, "Poster: Exploring the integrality and separability of the leap motion controller for direct manipulation 3d interaction," in *3D User Interfaces (3DUI), 2014 IEEE Symposium on*, pp. 153–154, IEEE, 2014.

[82] 3Dconnexion, "SpaceNavigator 3Dconnexion.." https://www.3dconnexion.eu/products/spacemouse/spacenavigator.html, 2017. Accessed: 2017-01-09.

[83] P. Cardoso, M. Melo, N. Gomes, A. Kehoe, and L. Morgado, "Adapting 3d controllers for use in virtual worlds," *DSAI*, pp. 27–30, 2007.

[84] M. Martins, A. Cunha, and L. Morgado, "Usability test of 3dconnexion 3d mice versus keyboard+ mouse in second life undertaken by people with motor disabilities due to medullary lesions," *Procedia Computer Science*, vol. 14, pp. 119–127, 2012.

[85] K. Gardström, "3d navigation for real-time mri using six degree of freedom interaction devices." Institutionen för teknik och naturvetenskap, 2003.

[86] "zSpace, inc.." https://zspace.com/. Accessed: 2017-01-09.

[87] C. Ware, K. Arthur, and K. S. Booth, "Fish tank virtual reality," in *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pp. 37–42, ACM, 1993.

[88] I. K. Li, E. M. Peek, B. C. Wünsche, and C. Lutteroth, "Enhancing 3d applications using stereoscopic 3d and motion parallax," in *Proceedings of the Thirteenth Australasian User Interface Conference-Volume 126*, pp. 59–68, Australian Computer Society, Inc., 2012.

[89] J. Vince, *Quaternions for computer graphics*. Springer Science & Business Media, 2011.

[90] D. A. Bowman and L. F. Hodges, "An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments," in *Proceedings of the 1997 symposium on Interactive 3D graphics*, pp. 35–ff, ACM, 1997.

[91] B. Shneiderman, "The future of interactive systems and the emergence of direct manipulation," *Behaviour & Information Technology*, vol. 1, no. 3, pp. 237–256, 1982.

[92] W. Javed, N. Elmqvist, J. S. Yi, *et al.*, "Direct manipulation through surrogate objects," in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 627–636, ACM, 2011.

[93] P. Saalfeld, J. Patzschke, and B. Preim, "An immersive system for exploring and measuring medical image data," *Mensch und Computer 2017-Tagungsband*, 2017.

[94] D. A. Bowman, E. Kruijff, J. J. LaViola Jr, and I. Poupyrev, "An introduction to 3-D user interface design," *Presence: Teleoperators and virtual environments*, vol. 10, no. 1, pp. 96–108, 2001.

[95] S. Feiner and A. Shamash, "Hybrid user interfaces: Breeding virtually bigger interfaces for physically smaller computers," in *Proceedings of the 4th annual ACM symposium on User interface software and technology*, pp. 9–17, ACM, 1991.

[96] I. G. Angus and H. A. Sowizral, "Embedding the 2D interaction metaphor in a real 3D virtual environment," in *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*, pp. 282–293, International Society for Optics and Photonics, 1995.

[97] M. Hachet, P. Guitton, and P. Reuter, "The CAT for efficient 2D and 3D interaction as an alternative to mouse adaptations," in *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 225–112, ACM, 2003.

[98] J. Wang and R. W. Lindeman, "Object impersonation: Towards effective interaction in tablet-and HMD-based hybrid virtual environments," in *Virtual Reality (VR), 2015 IEEE*, pp. 111–118, IEEE, 2015.

[99] M. M. Wloka and E. Greenfield, "The virtual tricorder: a uniform interface for virtual reality," in *Proceedings of the 8th annual ACM symposium on User interface and software technology*, pp. 39–40, ACM, 1995.

[100] K. Coninx, F. Van Reeth, and E. Flerackers, "A hybrid 2D/3D user interface for immersive object modeling," in *Computer Graphics International, 1997. Proceedings*, pp. 47–55, IEEE, 1997.

[101] J. Rekimoto, "Pick-and-drop: a direct manipulation technique for multiple computer environments," in *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pp. 31–39, ACM, 1997.

[102] B. Ullmer and H. Ishii, "The metaDESK: models and prototypes for tangible user interfaces," in *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pp. 223–232, ACM, 1997.

[103] S. Baumgärtner, A. Ebert, M. Deller, and S. Agne, "2D meets 3D: a human-centered interface for visual data exploration," in *CHI'07 Extended Abstracts on Human Factors in Computing Systems*, pp. 2273–2278, ACM, 2007.

[104] A. Bornik, R. Beichel, E. Kruijff, B. Reitinger, and D. Schmalstieg, "A hybrid user interface for manipulation of volumetric medical data," in *3D User Interfaces, 2006. 3DUI 2006. IEEE Symposium on*, pp. 29–36, IEEE, 2006.

[105] M. Teistler, R. Breiman, T. Lison, O. Bott, D. Pretschner, A. Aziz, and W. Nowinski, "Simplifying the exploration of volumetric images: development of a 3D user interface for the radiologists workplace," *Journal of digital imaging*, vol. 21, no. 1, pp. 2–12, 2008.

[106] M. Teistler, G. Ampanozi, W. Schweitzer, P. Flach, M. Thali, and L. Ebert, "Use of a low-cost three-dimensional gaming controller for forensic reconstruction of CT images," *Journal of Forensic Radiology and Imaging*, vol. 7, pp. 10–13, 2016.

[107] M. J. Graves, R. T. Black, and D. J. Lomas, "Constrained Surface Controllers for Three-dimensional Image Data Reformatting," *Radiology*, vol. 252, no. 1, pp. 218–224, 2009.

[108] R. P. Darken and R. Durost, "Mixed-dimension interaction in virtual environments," in *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 38–45, ACM, 2005.

[109] V. B. H. Mandalika, A. I. Chernoglazov, M. Billinghurst, C. Bartneck, M. A. Hurrell, N. de Ruiter, A. P. Butler, and P. H. Butler, "A hybrid 2d/3d user interface for radiological diagnosis," *Journal of Digital Imaging*, pp. 1–18, 2017.

[110] W. E. Brant and C. A. Helms, *Fundamentals of diagnostic radiology*. Lippincott Williams & Wilkins, 2012.

[111] R. A. Drebin, L. Carpenter, and P. Hanrahan, "Volume rendering," in *ACM Siggraph Computer Graphics*, vol. 22(4), pp. 65–74, ACM, 1988.

[112] "Human Ethics Committee university of canterbury." http://www.canterbury.ac.nz/humanethics/. Accessed: 2016-06-01.

[113] H. Hasche, R. Gockeln, and W. de Decker, "The titmus fly test–evaluation of subjective depth perception with a simple finger pointing trial. clinical study of 73 patients and probands," *Klinische Monatsblatter fur Augenheilkunde*, vol. 218, no. 1, pp. 38–43, 2001.

[114] R. Morrissy, G. Goldsmith, E. Hall, D. Kehl, and G. Cowie, "Measurement of the cobb angle on radiographs of patients who have," *J Bone Joint Surg Am*, vol. 72, no. 3, pp. 320–327, 1990.

[115] G. M. Cox and W. Cochran, *Experimental designs*. JSTOR, 1953.

[116] J. Brooke *et al.*, "SUS-A quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[117] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," *Journal of usability studies*, vol. 4, no. 3, pp. 114–123, 2009.

[118] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Intl. Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.

[119] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," *Advances in psychology*, vol. 52, pp. 139–183, 1988.

[120] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[121] S. E. Maxwell and H. D. Delaney, *Designing experiments and analyzing data: A model comparison perspective*, vol. 1. Psychology Press, 2004.

[122] E. Zudilova-Seinstra, B. van Schooten, A. Suinesiaputra, R. van der Geest, B. van Dijk, J. Reiber, and P. Sloot, "Exploring individual user differences in the 2D/3D interaction with medical image data," *Virtual Reality*, vol. 14, no. 2, pp. 105–118, 2010.

[123] R. E. Patterson, *Human factors of stereoscopic 3D displays*. Springer, 2015.

[124] J. Sweller, "Cognitive load during problem solving: Effects on learning," *Cognitive science*, vol. 12, no. 2, pp. 257–285, 1988.

[125] R. Aamir, A. Chernoglazov, C. Bateman, A. Butler, P. Butler, N. Anderson, S. Bell, R. Panta, J. Healy, J. Mohr, *et al.*, "MARS spectral molecular imaging of lamb tissue: data collection and image analysis," *Journal of Instrumentation*, vol. 9, no. 02, p. P02005, 2014.

[126] M. Walsh, S. Nik, S. Procz, M. Pichotka, S. Bell, C. Bateman, R. Doesburg, N. De Ruiter, A. Chernoglazov, R. Panta, *et al.*, "Spectral ct data acquisition with medipix3. 1," *Journal of Instrumentation*, vol. 8, no. 10, p. P10012, 2013.

[127] A. Atharifard, J. Healy, B. Goulter, M. Ramyar, L. V. Broeke, M. Walsh, C. Onyema, R. Panta, R. Aamir, D. Smithies, *et al.*, "Per-pixel energy calibration of photon counting detectors," *Journal of Instrumentation*, vol. 12, no. 03, p. C03085, 2017.

[128] L. V. Broeke, A. Atharifard, B. Goulter, J. Healy, M. Ramyar, R. Panta, M. Anjomrouz, M. Shamshad, A. Largeau, K. Mueller, *et al.*, "Oblique fluorescence in a mars scanner with a cdte-medipix3rx," *Journal of Instrumentation*, vol. 11, no. 12, p. C12063, 2016.

[129] S. Frees and G. D. Kessler, "Precise and rapid interaction through scaled manipulation in immersive virtual environments," in *IEEE Proceedings. VR 2005. Virtual Reality, 2005.*, pp. 99–106, IEEE, 2005.

Appendix A

Forms and Questionnaires

A.1 Human ethics approval form

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

HUMAN ETHICS COMMITTEE

Secretary, Rebecca Robinson
Telephone: +64 03 364 2987, Extn 45588
Email: human-ethics@canterbury.ac.nz

Ref: HEC 2016/35/LR-PS

30 June 2016

Veera Bhadra Harish Mandalika
HIT Lab NZ
UNIVERSITY OF CANTERBURY

Dear Veera Bhadra Harish

Thank you for submitting your low risk application to the Human Ethics Committee for the research proposal titled "Evaluation of a 2D/3D Hybrid User Interface for Medical Diagnosis".

I am pleased to advise that this application has been reviewed and approved.

Please note that this approval is subject to the incorporation of the amendments you have provided in your email of 22nd June 2016.

With best wishes for your project.

Yours sincerely

pp. R. Robinson

Jane Maidment
Chair, Human Ethics Committee

A.2 Information Sheet

User study to evaluate a hybrid 2D+3D user interface for medical visualization

Information Sheet

Hi, my name is Harish Mandalika. I am a PhD student at the HIT Lab, University of Canterbury. You are invited to take part in this user study which will form a part of my doctoral thesis. The purpose of this study is to evaluate a novel hybrid (2D+3D) user interface to interact with volumetric medical data (CT scan data). This study will contribute to the ongoing research in design and use of hybrid user interfaces for medical visualization.

Risks / Discomforts

Risks are minimal in this study. The participants will be seated and perform simple diagnosis tasks using mouse and keyboard. The participants will wear passive stereo glasses similar to the ones worn in cinemas to watch a 3D movie and use a 3D stylus for interaction.

Confidentiality

All data obtained from participants will be kept confidential. The data will be kept securely for a period of 10 years and will be destroyed after that period. In publications and PhD thesis, the results will be reported in an aggregate form. These documents will be available publicly through the UC Library.

Participation

Participation in this research study is completely voluntary. You have the right to withdraw at any time or refuse to participate entirely.

Study conducted in collaboration with HIT Lab NZ, University of Canterbury (UC), Canterbury District Health Board (CDHB), University of Otago, Christchurch (UOC) and MARS Bioimaging Ltd.

Video Recording

Video of the experiment will be recorded for analysis. The video will only contain the workspace and the participant's hand. The intent is to record the stylus interaction during the task. The participant's face will not be recorded to maintain the anonymity of the participant. Only the researcher and supervisors will have access to the video recordings. Recordings will be kept securely for a period of 10 years and then they will be destroyed.

Approval for this study

This study has been approved by University of Canterbury Human Ethics Committee Low Risk Approval process.

Questions

If you have any further questions regarding this study, you may contact the researcher

- Harish Mandalika (harish.mandalika@pg.canterbury.ac.nz) or supervisors
- Christoph Bartneck (Christoph.bartneck@canterbury.ac.nz)
- Niels de Ruiter (niels.deruiter@canterbury.ac.nz).

Please take this information sheet with you when you leave.

Study conducted in collaboration with HIT Lab NZ, University of Canterbury (UC), Canterbury District Health Board (CDHB), University of Otago, Christchurch (UOC) and MARS Bioimaging Ltd.

A.3 Start questionnaire

Start Questionnaire

Participant ID : _____ Task Number #: _____ Condition : _____

1. Gender : [ ] Male [ ] Female
2. Age : [ ]
3. Occupation :
   [ ] Medical Student
   [ ] Registrar
   [ ] Other : _____(please specify)
4. How experienced with computers do you rate yourself?
   (1: Novice ~ 4: Moderate ~7: Expert)
   1 2 3 4 5 6 7
5. Have you used Inteliviewer software before? [ ] Yes [ ] No
6. If Yes, how frequently do you use it.
   [ ] daily [ ] once a week [ ] once a month [ ] once a year
7. Have you used any other similar medical diagnosis or CT viewing software?
   [ ] Yes [ ] No
   please specify the name : _____
8. If Yes, how frequently do you use it.
   [ ] daily [ ] once a week [ ] once a month [ ] once a year
9. Approximately how many 3D movies have you watched at Cinemas in the past two years. (Please specify 0 if you haven't watched any) :
   _____
10. Do you own a 3DTV or a 3D projector at home? [ ] Yes [ ] No
11. If Yes, How frequently do you watch 3D content on it.
    [ ] daily [ ] once a week [ ] once a month [ ] once a year
12. Do you own a computer with a 3D capable Monitor? [ ] Yes [ ] No
13. If Yes, How frequently do you watch 3D content or play Games in 3D.
    [ ] daily [ ] once a week [ ] once a month [ ] once a year
14. Have you ever used a zSpace interface before? [ ] Yes [ ] No
15. If Yes, How frequently do you use zSpace.
    [ ] daily [ ] once a week [ ] once a month [ ] once a year

16. Do you own any other 3D interface?
    [ ] Yes, Specify : _____
    [ ] No
17. If Yes, How frequently do you watch 3D content on it.
    [ ] daily [ ] once a week [ ] once a month [ ] once a year
18. Do you wear prescription glasses? [ ] Yes [ ] No
19. How frequently do you use the scroll wheel on a computer mouse? Circle your choice.
    1 2 3 4 5

    Very Frequently Very Rarely

20. Have you ever used a mouse with a middle click / scroll click? [ ] Yes [ ] No
21. If Yes, How frequently do you use the middle click / scroll click? Circle your choice
    1 2 3 4 5

    Very Frequently Very Rarely

A.4 End questionnaire

End Questionnaire

Participant ID : _____ Task Number #: _____ Condition : _____

1. Rate the interfaces for the overall task. Circle your choice.

	Best				Worst
a. 2D -	1	2	3	4	5
b. 3D -	1	2	3	4	5
c. 5D -	1	2	3	4	5

2. How easy was the overall diagnosis task? Circle your choice.

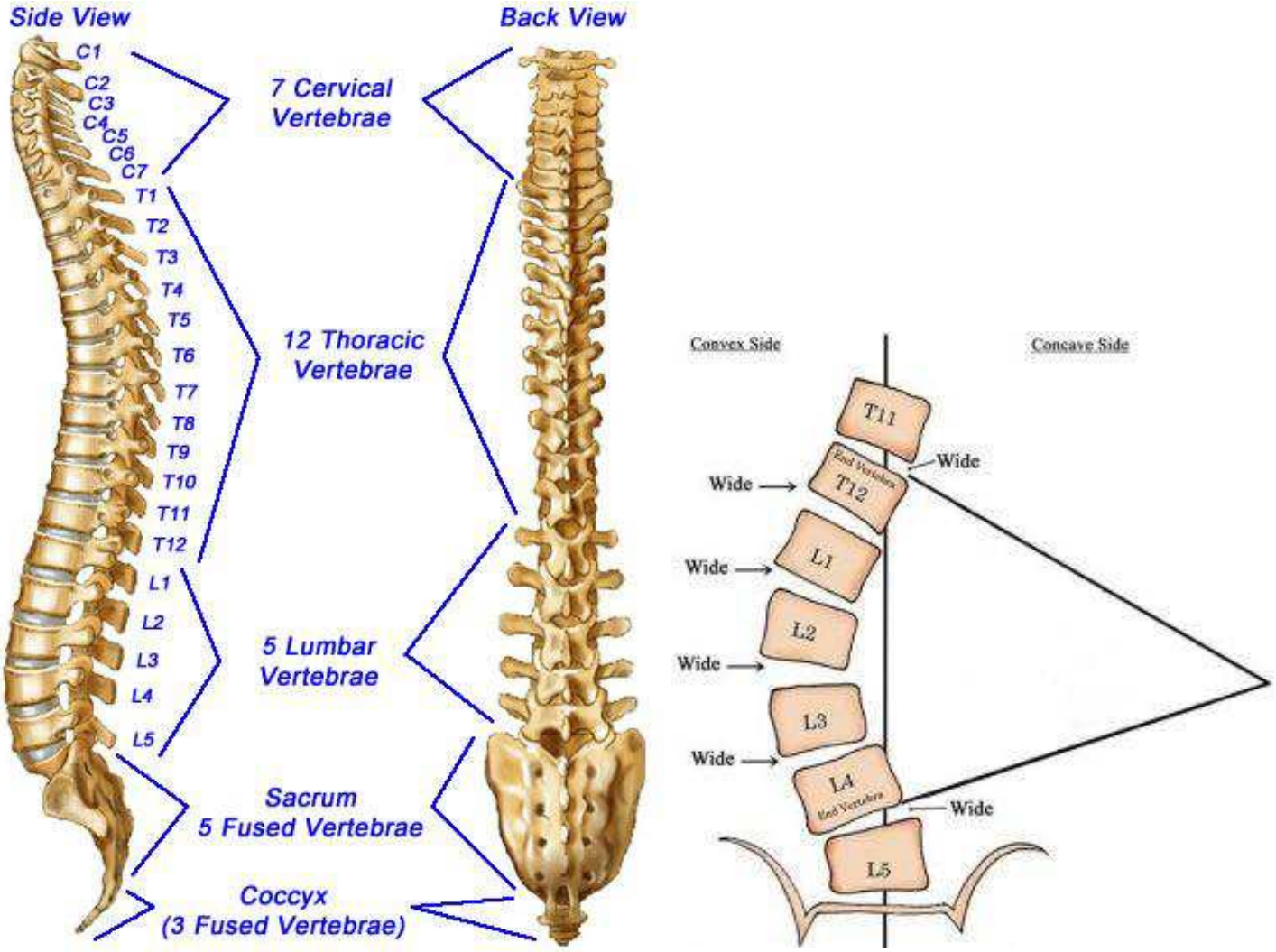
1	2	3	4	5	6	7
Very Hard						Very Easy

3. Please list any particular problems you had while performing the tasks using any of the interfaces. _____

Appendix B

Reference Cards

B.1 Spinal column



B.2 2D Controls

Condition – 2D

Slice View Manipulation

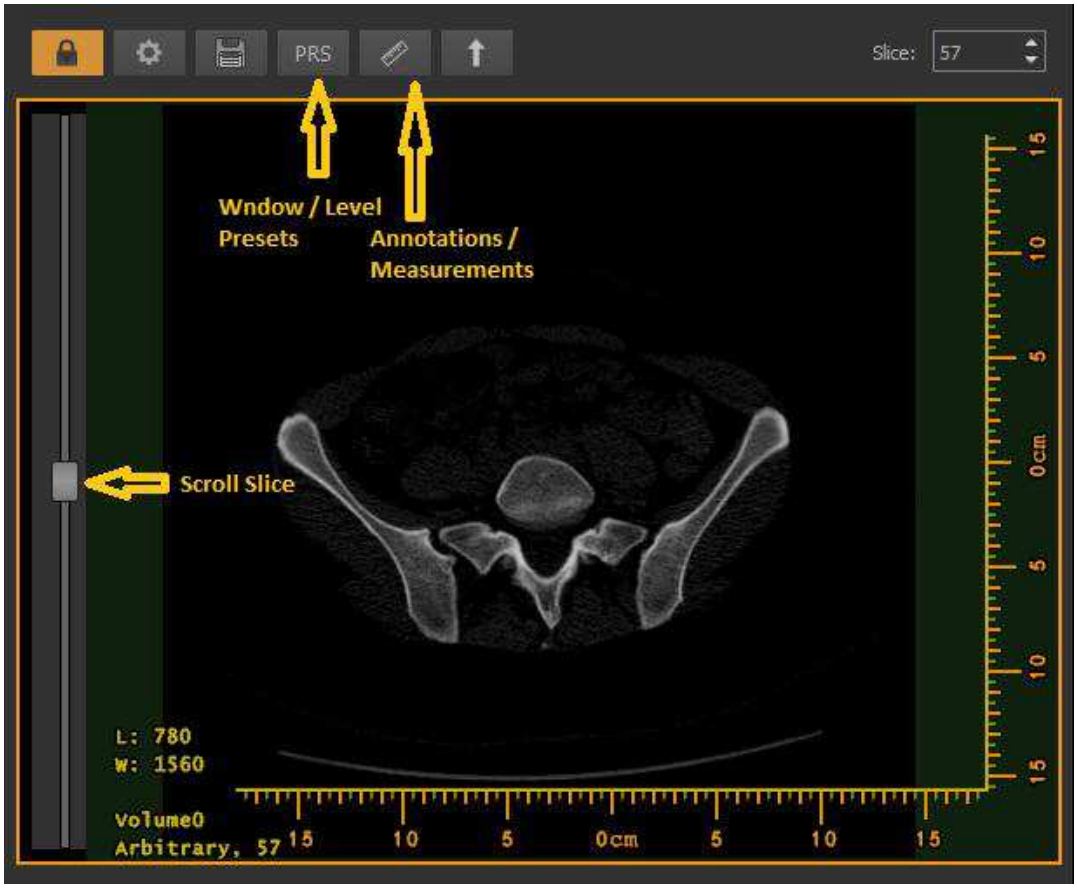
View pan	Left mouse click + drag mouse
View zoom	Hold [Shift] key + scroll wheel
View scroll (scroll slices)	Scroll wheel
View window/level adjust	Hold middle mouse Button + drag mouse
Rotate slice (Arbitrary view only)	Hold [Shift] key + left click + drag mouse

Annotations

Place annotation	Hold [Ctrl] key + Left Click
Select annotation	Right click near annotation
Delete selected annotation	Press [Delete] key

Line / Angle Measurements

Draw line	Right mouse click + drag
Select line	Right mouse click (end points)
Delete line	Press [Delete] key
Select lines for angle measurement	Right mouse click (on lines)



B.3 3D Controls

Condition – 3D

Slice Manipulation – (Toggle slice first)

Slice manipulation	Stylus front button
Slice translation (no rotation)	Stylus right button

Annotations

Place annotation	Stylus right button
Select annotation	Simply point the stylus near the point
Move annotation	Stylus front button
Delete annotation	Stylus left button

Line Measurements

Place a line point	Stylus right button
Select line point	Simply point the stylus near the point
Move line points	Stylus front button
Delete line (select either point)	Stylus left button

Angle Measurements

Select a Line (select its endpoint)	Stylus right button
Unselect all lines	Stylus left button
Swap endpoints (once 2 lines are selected)	Stylus right button

B.4 Hybrid Controls

Condition – 5D (2D + 3D)

Slice Manipulation – (Toggle slice first)

Slice manipulation	Stylus front button
Slice translation (no rotation)	Stylus right button

Annotations

Place annotation	Hold [Ctrl] key + Left Click
Select annotation	Right Click near Annotation
Delete selected annotation	Press [Delete] key

Line / Angle Measurements

Draw Line	Right mouse click + drag
Select Line	Right mouse click (end points)
Delete Line	Press [Delete] key
Select Lines for Angle Measurement	Right mouse click (on lines)